13 SOLID STATE ELECTRONICS

Field-effect transistors

Introducing FETs

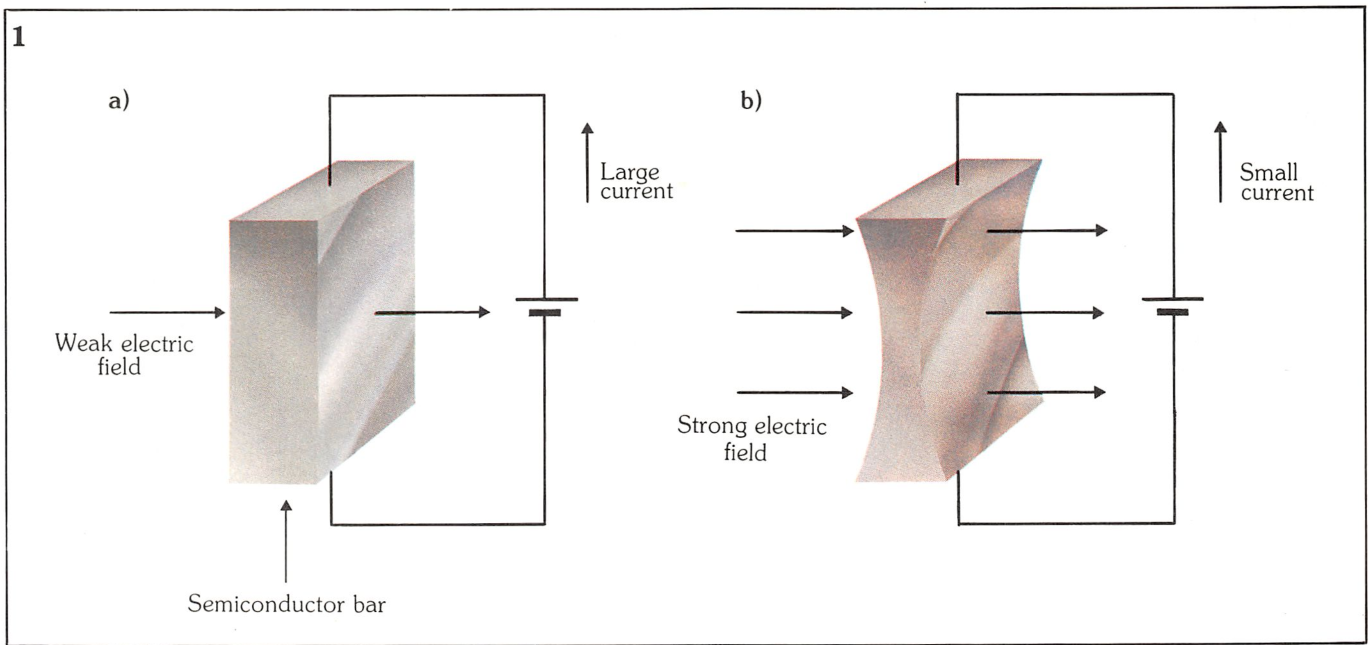
1. **(a) A weak perpendicular electric field** applied to the semiconductor bar; **(b)** a strong electric field has the effect of squeezing the semiconductor bar, increasing its resistance thereby reducing the current flow.

By now, you will be familiar with bipolar transistors which are essentially current operated amplifiers. In the n-p-n bipolar transistor, for example, a small current applied into the base (the base current) produces a larger current flow into the collector (the collector current); similarly, base current taken from the base of a p-n-p bipolar transistor produces a larger current flow from the collector.

Although bipolar transistors have now been produced for about 35 years, their development was actually preceded by that of the field-effect transistor or FET, which has only recently been manufactured on a commercial basis. It was while investigating manufacturing problems that the bipolar transistor effect was discovered; being simpler to manufacture, they were produced first.

There are two types of field-effect transistor: the first is the **insulated gate FET**, also known as an IGFET, in which a thin film of semiconductor and a metal plate are formed into a capacitor. Application of a voltage across this capacitor creates a conduction path along the semiconductor surface. Modern derivatives of the FET include MOSFETs and CMOS ICs. The second type of FET relies on the conduction path being made *through* the semiconductor material. As operation of this device depends upon the action of a reverse-biased p-n junction, it is known as a **junction field-effect transistor** (JFET).

1

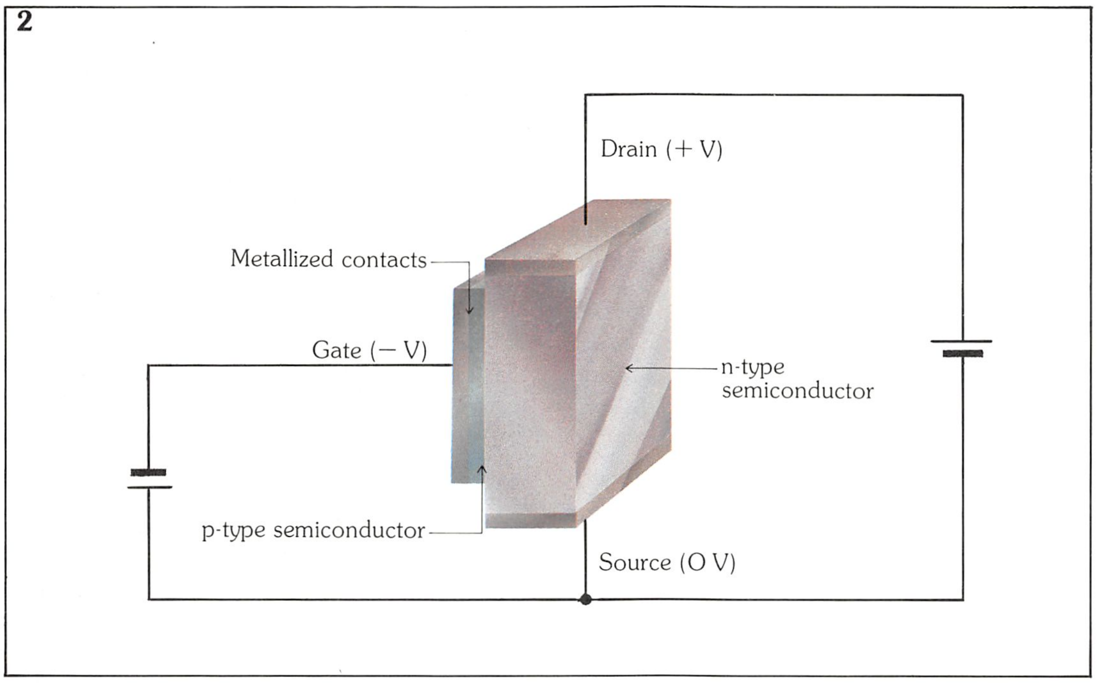


Field-effect transistors differ from junction transistors in that they are unipolar (i.e. current is carried by majority carriers only) and conduction occurs through the substrate. Junction transistors, on the other hand, are bipolar (i.e. current is carried by both majority and minority carriers) and conduction occurs across junctions.

FET action

A FET can be thought of as a bar of

2



2. Simplified structure of a FET showing gate, source and drain.

3. A more typical structure of a FET.

4. Cross-sectional diagrams and circuit symbols for:
(a) n-channel JFET;
(b) p-channel JFET.

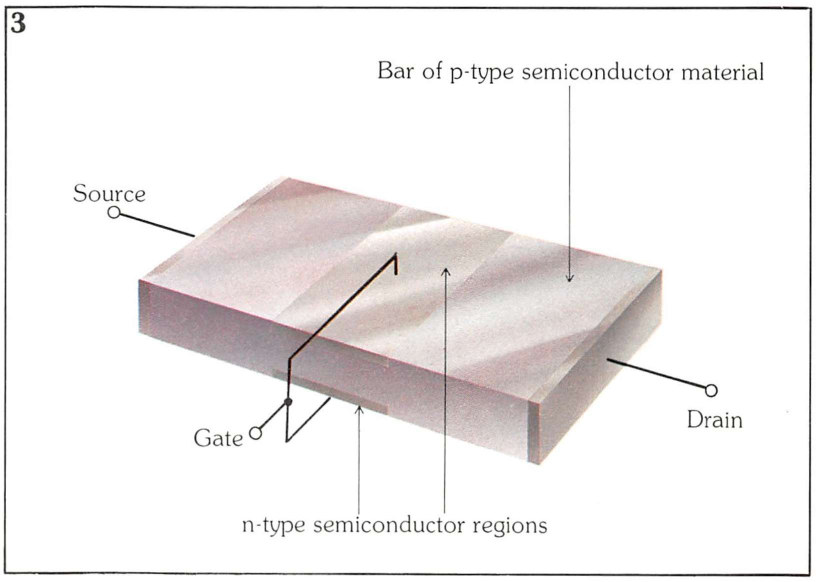
5. A p-channel JFET with a positive voltage applied to the gate.
(a) the space-charge region is fairly small; **(b)** the two space-charge regions meet.

semiconductor material through which a perpendicular electric field is applied (*figure 1a*). The strength of this electric field defines the resistance of the semiconductor bar: a strong electric field (*figure 1b*) produces a high resistance; a weak field has no effect, so the resistance is low. If a voltage is applied across the semiconductor bar, the current flowing through it can be *controlled* by the perpendicular electric field in this way (hence the name field-effect). A general concept of FET operation is that the resistance of the bar is reduced by reducing the number of charge carriers available for conduction.

Say, as in *figure 2*, the semiconductor bar is composed of n-type material, adding p-type material onto its side forms a p-n junction. The p-type semiconductor is called the **gate**. Metallizing the two ends of the semiconductor bar allows conducting leads to be attached: these contacts are known as the **source** and **drain**. These three terminals, source, drain and gate, correspond to the emitter, collector and base respectively of a bipolar transistor.

From the voltages shown connected to the FET in *figure 2*, you can see that the drain is positive with respect to the source and so current can flow across the bar. If the gate is made negative we can see that the p-n junction formed by the bar and the gate is reverse-biased.

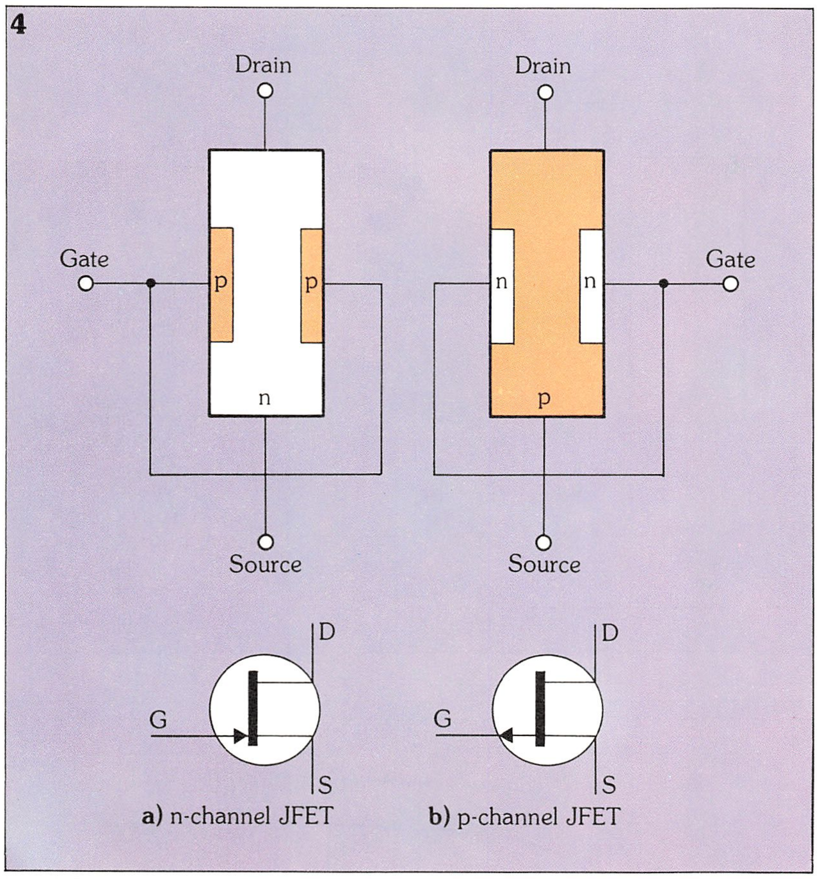
3


Bar of p-type semiconductor material
Source
Gate
Drain
n-type semiconductor regions

When the voltage between the gate and the source, V_{GS}, is 0 V, then the electric field perpendicular to the semiconductor bar is at its weakest, therefore the bar's resistance is at its lowest (about 100 Ω). As the negative voltage is increased, the electric field increases until the bar ultimately ceases to conduct (a resistance as high as 10 MΩ) and appears like an open circuit between source and drain. At intermediate gate-source voltages the drain-source resistance varies between the two limits.

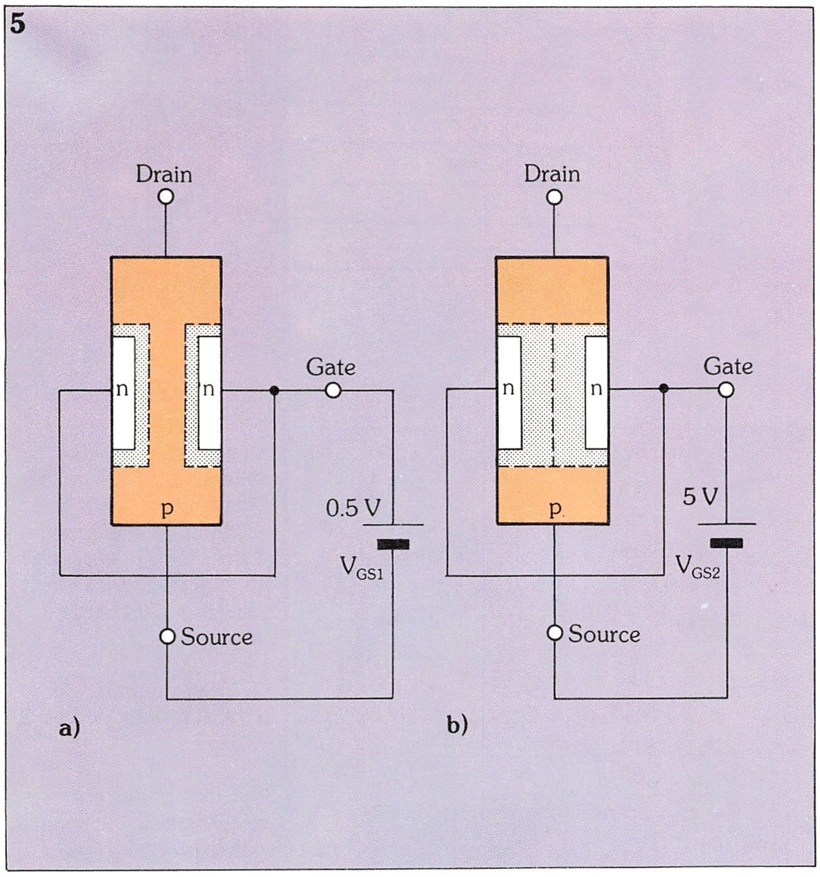
A more typical design for a JFET is

4



a) n-channel JFET **b)** p-channel JFET

5



shown in *figure 3* where n-type semiconductor regions are on opposing surfaces of the p-type semiconductor bar, but are electrically connected. The interesting part to note is the region between the two p-n junctions. We have previously discussed the existence of space-charge regions (where there are no charge carriers) at p-n junctions. If the space-charge regions at each p-n junction in *figure 3* are sufficiently wide to meet in the middle of the semiconductor bar, then no current will flow, i.e. the bar's resistance is high. Conversely, narrow space-charge regions do allow a flow of charge carriers producing a low resistance in the bar.

As the width of the space-charge region at a p-n junction increases as a reverse-bias voltage is applied, you can see that the applied reverse-bias voltage, V_{GS}, controls the conductance of the bar.

The conduction path between drain and source is called the **channel**, and as the bar in this example is made of p-type material, this transistor is a **p-channel JFET**; n-channel JFETs are also produced but, as with n-p-n and p-n-p transistors, voltages and currents are reversed.

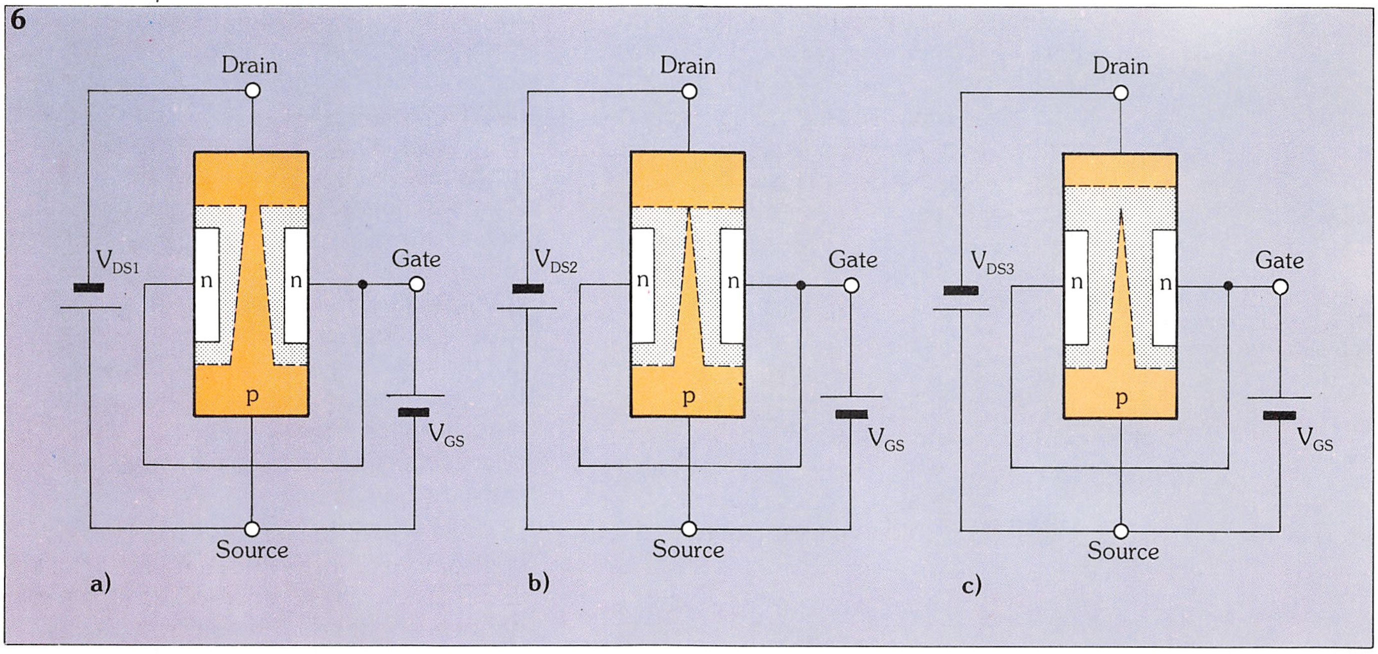
The JFET is a symmetrical component and drain and source terminals are interchangeable: they can only be distinguished by the direction of current in the bar. Therefore, the terminal from which the charge carriers come is considered the source, and the terminal to which the charge carriers go is the drain.

Figure 4 illustrates the design of circuit symbols for both n-channel (*figure 4a*) and p-channel (*figure 4b*) types. Note that the arrows point from the p-type doped region to the n-type doped region.

Electrical operation

If the drain of a JFET is left electrically unconnected but the gate is reverse-biased with respect to the source, then the effect shown in *figure 5* occurs. A p-channel JFET is illustrated with a positive voltage applied to the gate. The space-charge region (indicated by the broken lines) is shown in *figure 5a* to be fairly small. This might occur if V_{GS1} is small, say, 0.5 V. However, as the gate-source voltage is increased to, say, 5 V as in *figure 5b*, then the two space-charge regions meet. At this

6

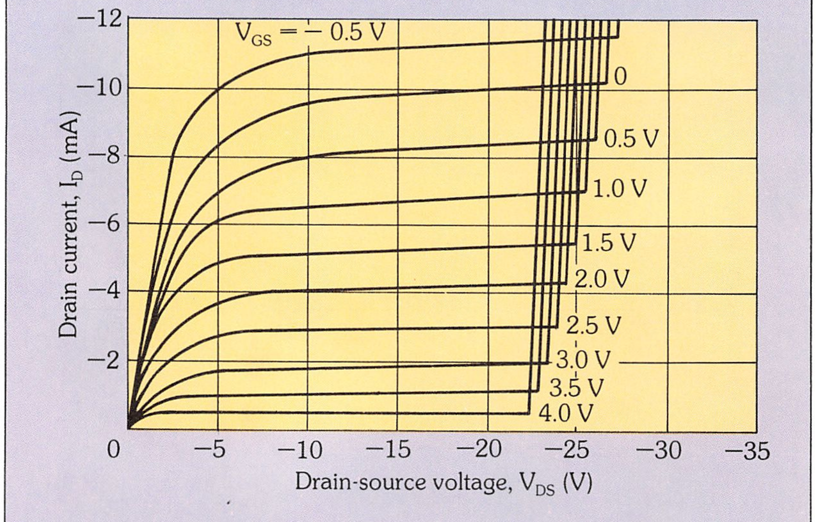


point, no charge carriers can exist and the resistance between drain and source is very high (in the order of megohms). The JFET is switched off, and the voltage between gate and source at which this occurs is known as the **pinch-off voltage**, V_p.

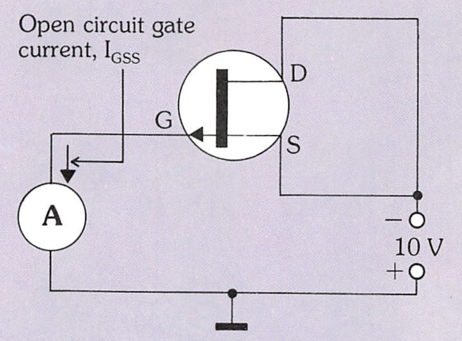
In *figure 6*, a constant voltage less than the above pinch-off voltage has been applied across gate and source, and three different values of drain-source voltage (V_{DS}) are shown, together with their effect on the space-charge regions. With a low drain-source voltage, V_{DS1} in *figure 6a*, the space-charge regions do not meet. Now, the flow of charge carriers through the channel from source to drain produces a voltage drop throughout the length of the channel. So at one point along the channel the reverse-bias voltage V_{GS} might be, say 0.1 V, further down the channel V_{GS} might be 0.2 V and so on. In a p-channel JFET, the drain voltage is negative with respect to that of the source and so higher reverse-bias voltages are experienced by the p-n junctions the closer they are to the drain. The space-charge regions are thus formed into wedge-shaped zones which are thickest at the drain end of the channel. A bottleneck is created which restricts the flow of charge carriers from source to drain.

In *figure 6b* the drain-source voltage has increased to V_{DS2} and the space-charge regions have widened towards the

7



8



6. p-channel JFETs where a constant voltage is applied between gate and source. Increasing values for V_{DS} are shown together with their effect on the space-charge region.

7. The principal characteristic curves of a JFET.

8. Circuit diagram showing how the reverse gate current, I_{GSS}, is measured.

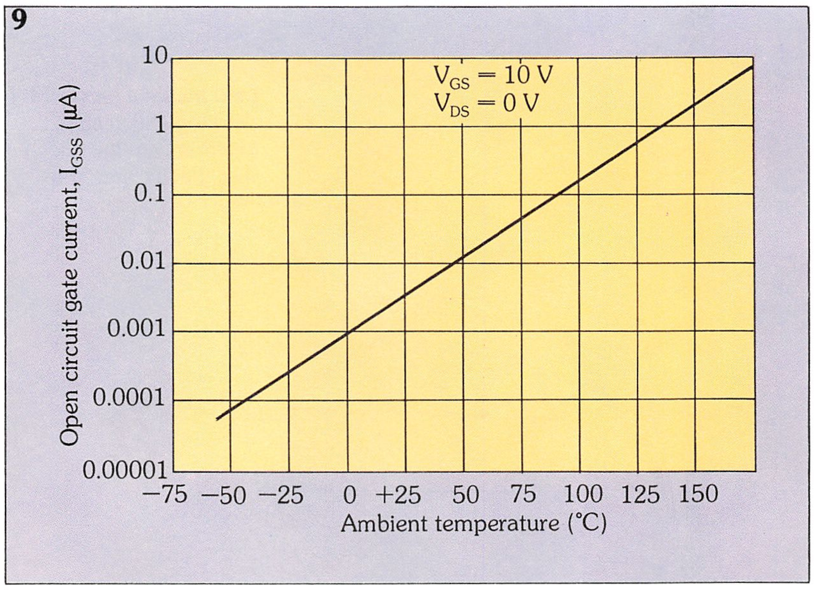
drain until they provoke a pinch-off situation, but the gate-source voltage is *less than the above pinch-off voltage, V_p.*

If the drain-source voltage increases further, as in *figure 6c*, the channel remains more or less the same shape but the space-charge region extends towards the drain. The effect of this triangular shaped channel between two wedge-shaped space-charge regions is that the resistance of the channel is not as high as that obtained in the circuit of *figure 5*. A number of charge carriers will therefore be able to flow from source to drain, creating a drain current I_D, which remains constant even if the gate-source voltage increases. The JFET is said to be **saturated** and, at this value of V_{DS}, can never be turned *totally* off.

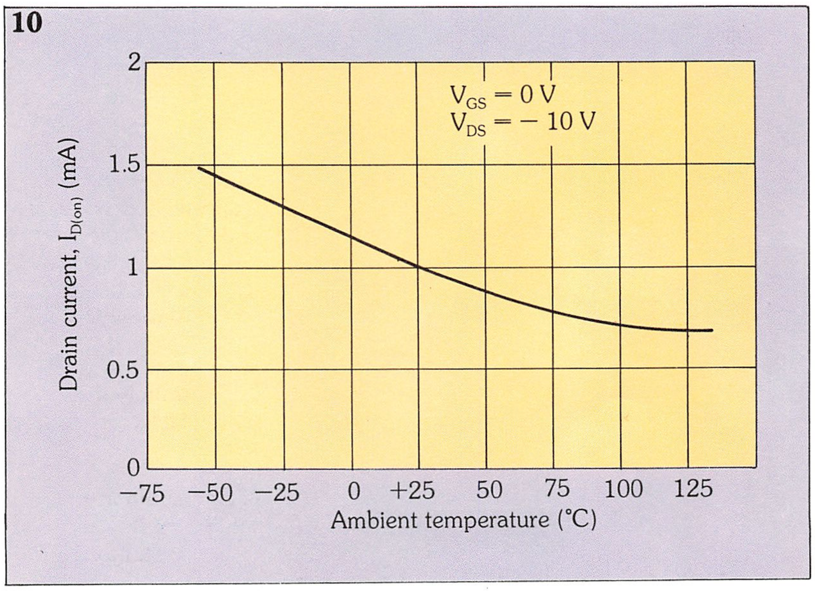
9. I_{GSS} plotted against temperature for a silicon JFET.

10. JFET characteristic curve showing drain current to be inversely proportional to temperature.

9



10



JFET characteristics

The principal characteristic curves of a JFET – the output curves – in which drain current is plotted against drain-source voltage are shown in *figure 7*. Various gate-source voltages are used as reference parameters. These curves have two areas of interest: the initial steep incline where the drain current is dependent on the drain-source voltage; and the levelling off where the drain current is almost constant.

Where the curve is steep, the channel functions in a virtually linear way as a simple resistor. In fact, it is such a good resistor that Ohm's law can be applied to a JFET as long as the pinch-off voltage has not been reached. The output characteristic of a JFET can be used to calculate the **differential drain resistance** r_d, which is the ratio between the variations of drain-source voltages and the corresponding variations in drain current. Typical values are around 10 kΩ.

Reverse gate current

Connecting a JFET drain to its source and reverse-biasing the gate-channel p-n junction, as shown in *figure 8*, allows the **reverse gate current**, I_{GSS}, to be measured. *Figure 9* shows the typical variation of this current with temperature, for a silicon JFET. From the graph it can be seen that the input resistance is about 1×10^{10} Ω at 0° C, falling at about 50 MΩ at 100 °C.

Drain current and temperature

Figure 10 shows a characteristic curve of a JFET, of the drain current for a given drain-source voltage when the transistor is operated in its pinch-off region. In the temperature range shown, the drain current is seen to be inversely proportional to temperature – contrary to the collector currents of bipolar transistors.

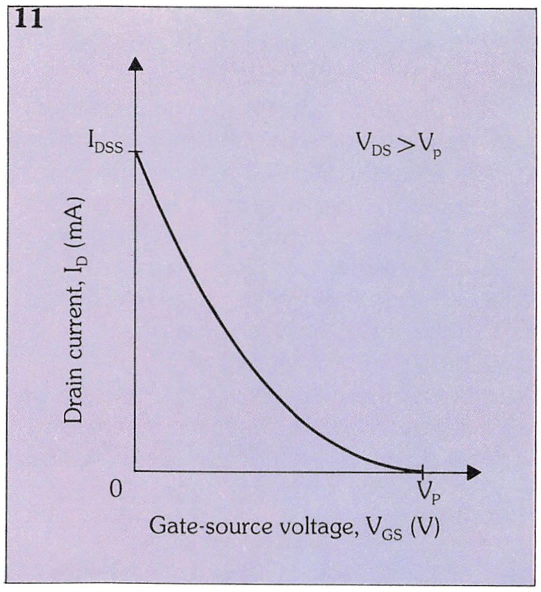
Drain current in the pinch-off region, when measured for a given drain-source voltage, is called the **on-state drain current**, $I_{D(on)}$. With no bias on the gate-source p-n junction, the drain current is given the symbol I_{DSS}. If the output characteristic curves are flat in the pinch-off region, I_{DSS} approximates to $I_{D(on)}$.

Transfer characteristic

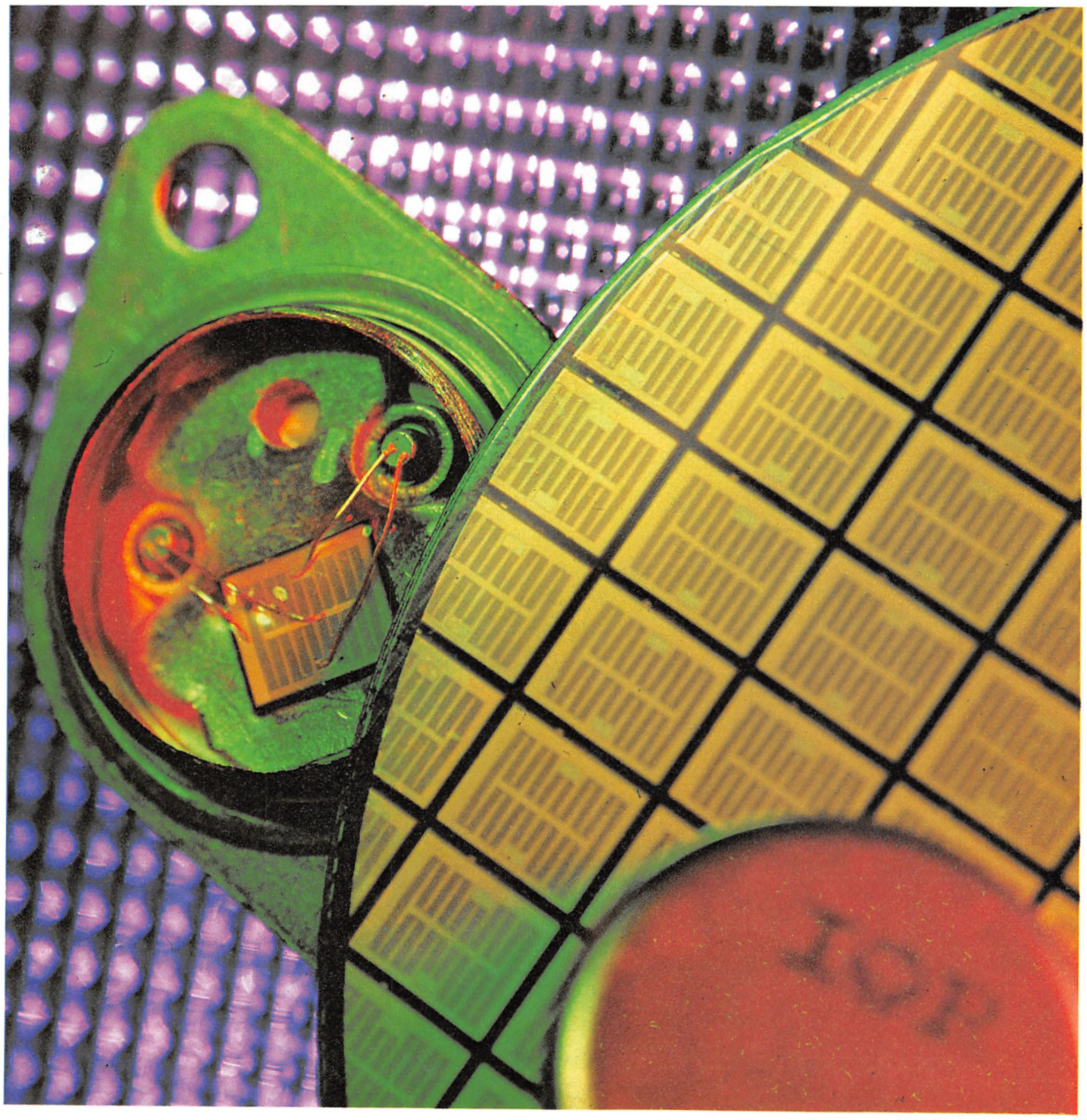
The final JFET curve we shall look at is shown in *figure 11*, the **mutual** or **transfer characteristic**. The curve shows the gate-source voltage against the drain current, when the drain-source voltage is higher than the pinch-off voltage. Drawing the curve when V_{DS} is greater than V_p ensures that the device is saturated and so the drain-source voltage has little or no influence over the drain current. In this way, a single curve (rather than a family of curves) is sufficient.

The **transconductance**, g_m, of a JFET can be calculated from the transfer characteristic curve as the ratio between a variation of drain current and the corresponding variation of V_{GS}.

11



11. The mutual or transfer characteristic of a JFET.



Left: inside a power FET showing silicon chip, together with the silicon slice that it came from.

Paul Brierley

The transfer characteristic has an almost parabolic shape which can be expressed by the formula:

$$I_D = \left(1 - \frac{V_{GS}}{V_p}\right)^2 I_{DSS}$$

The differential drain resistance, r_d, and the transconductance, g_m, are linked together through the amplification coefficient, μ, by the expression:

$$\mu = r_d \times g_m$$

In this way, if we know two of the parameters, the third can be calculated. The value of the amplification coefficient can be taken from the output characteristic curves.

Uses of a JFET

The characteristic curves of a JFET show how the output current is controlled by input voltage. Because the input resistance of a JFET is extremely high, virtually no gate current flows. If we compare this with a bipolar transistor the main difference is obvious – a bipolar transistor needs base current to operate; a JFET is voltage operated.

A JFET is therefore turned on and off by application or removal of voltage, forming, in simplest terms, a voltage-controlled switch.

If the output current is passed through a resistance, the voltage developed across the resistance may well be larger than the input voltage, and the JFET is therefore turned into a voltage-controlled amplifier.

When operated at very low values of V_{GS} (of the order of 0.5 V), a JFET is essentially a voltage-controlled resistor between source and drain, which closely follows Ohm's law, and can be used in circuits to form a voltage-controlled variable resistor.

Finally, the extremely high input resistance of a JFET means that it can be used as the basis of a buffer amplifier, interfacing circuits which only develop small output currents to circuits which require large input currents.

Glossary

channel	the conduction path between drain and source of a FET, made of a single piece of doped semiconductor – either n-type or p-type
FET saturation	effect in a FET, when pinch-off is reached, and drain current is constant but independent of drain-source voltage
field-effect	a phenomenon which occurs to a single piece of doped semiconductor, in which the resistance varies as a perpendicular electric field is applied
transconductance, g_m	ratio between variations of drain current and the corresponding variations of gate-source voltage. Can be calculated from the transfer characteristic of a FET
n-channel FET	a FET in which the channel is made from a piece of n-type semiconductor
p-channel FET	a FET, the channel of which is made from p-type semiconductor
pinch-off voltage, V_p	the gate-source voltage, above which a constant drain current flows
transfer characteristic	characteristic curve of a FET, relating gate-source voltage to drain current

ELECTRICAL TECHNOLOGY

Electromagnetic induction

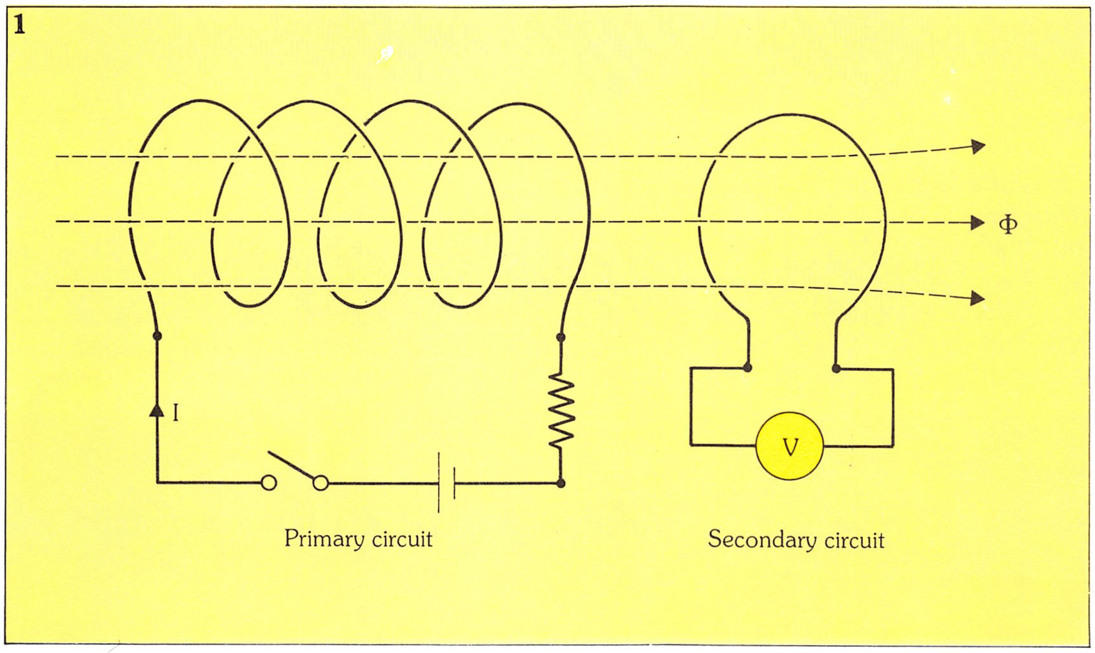
So far we have only looked at the constant fields produced around stationary conductors, when constant currents flow through them. But such systems are of little use to us in any real application, apart from, say, electromagnets. In this chapter, we'll move on to look at a phenomenon which has far-reaching effects in the world of electronics.

Induction

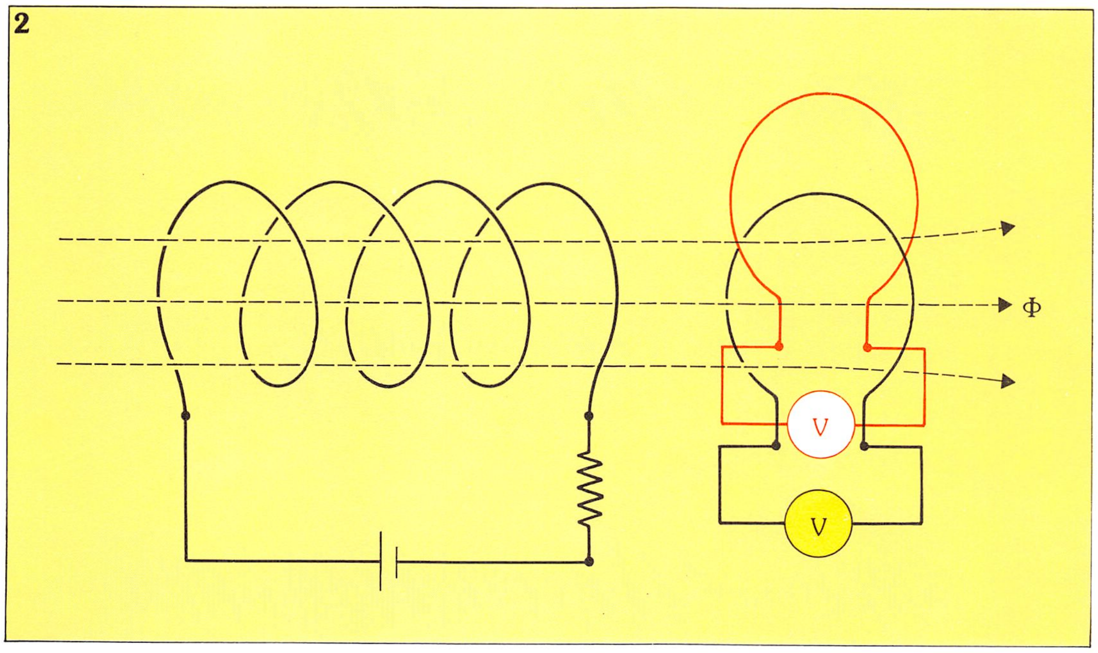
In 1831, Michael Faraday discovered that turning the current in a coil of wire on or off, induced an EMF in an adjacent wire (see *figure 1*). However, while the current is flowing in the **primary** coil no EMF is produced in the **secondary** wire. This phenomenon has since been named **electromagnetic induction**. An interesting point to note is that when the switch in a primary circuit, such as that in *figure 1*, is closed, the induced EMF is in the opposite direction to that when the switch is open.

The same sort of effect happens if a

1



2



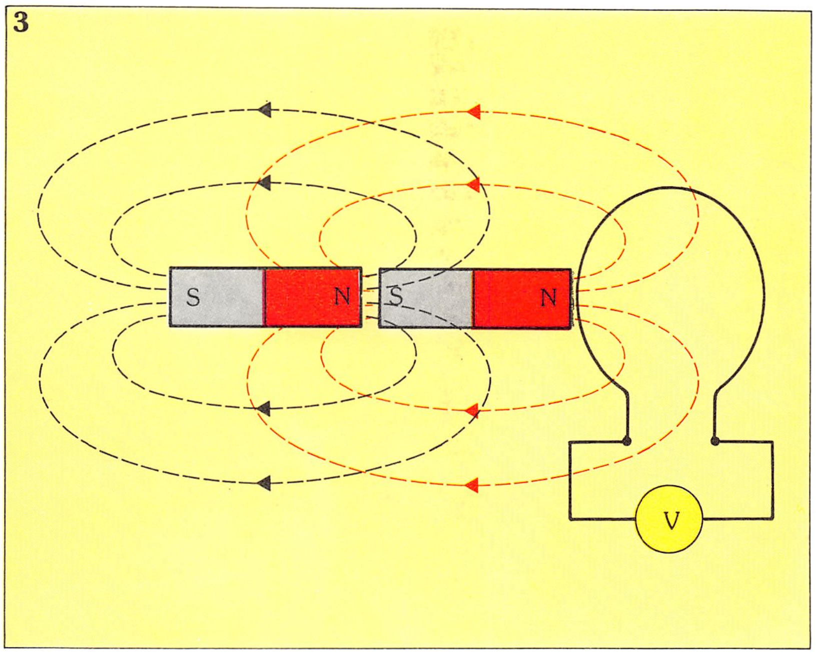
1. Turning the current in a coil of wire on and off induces an EMF in an adjacent wire.

2. A similar effect to that shown in figure 1: a steady current flows in the primary circuit and if the secondary circuit is suddenly moved to a new position, then an EMF is induced.

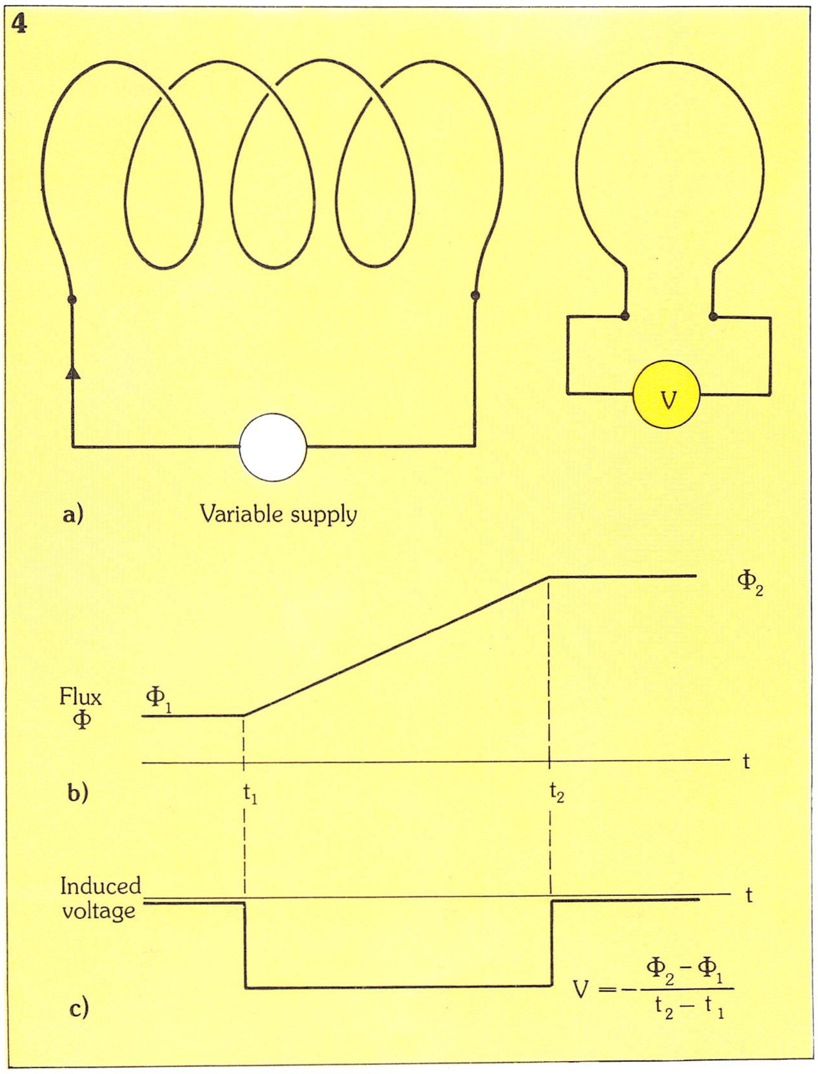
3. Plunging a bar magnet into a secondary wire loop also includes an EMF.

4. (a) A variable supply connected to a primary coil causes flux linkages to occur within the secondary; **(b)** these change from ϕ_1 at t_1 to ϕ_2 at t_2; **(c)** the rate of change of flux linked gives the magnitude of the induced EMF.

3



4



steady current flows in a primary circuit and the secondary circuit is suddenly moved to a new position, as the black and red secondaries in *figure 2* indicate. While the secondary wire is actually in motion, an EMF is induced and can be seen on the voltmeter; but as soon as the secondary stops, the EMF disappears. Furthermore, in the same way that EMF polarity depends on whether the current is turned on or off, so the polarity depends in which direction the primary moves – away from, or towards the secondary.

This effect also occurs if a bar magnet is suddenly plunged into a secondary wire loop, as indicated by the black and red magnets in *figure 3*. While the magnet is in motion, an EMF is induced in the secondary, but as soon as the magnet stops moving the EMF falls to zero. Again, the direction of the induced EMF depends on whether the magnet moves towards or away from the secondary coil.

In these three examples we have suggested that the movement or change of current should be performed suddenly – but this is not essential. We could replace the battery and switch in *figure 1* by a variable source of voltage. In this case, we would find that the voltmeter shows a reading whenever the primary voltage changes. Perhaps of more importance, the induced secondary EMF is greater when the primary voltage changes more quickly.

Similarly, the induced secondary EMF in *figures 2* and *3* will be greater when movement of the coil or magnet is faster.

Changing flux

All three effects can be explained by a single concept: the induced secondary EMF is generated by the change of flux around the primary (whether the primary is a current-carrying coil or a magnet).

We can go further than this simple statement and define the value of the induced EMF, because the **magnitude** of the induced EMF (in V) in a single turn of wire is equal to the rate of change of flux linking with the turn with time (in Wb s^{-1}).

Consider the circuit in *figure 4a*. The variable supply connected to the primary coil causes flux linkages to occur within the secondary, which changes from a value ϕ_1 Wb at time t_1, to a value ϕ_2 Wb at time t_2. This is illustrated in *figure 4b*. The change of flux linked is given by:

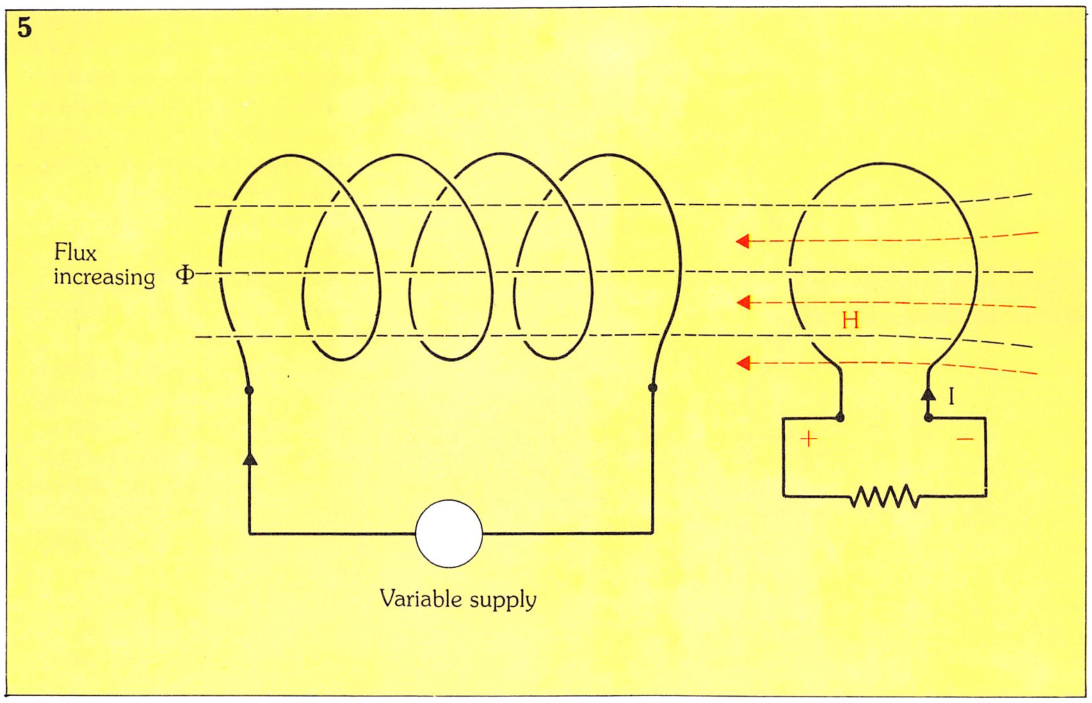
$$\phi_2 - \phi_1$$

over a period of time:

$$t_2 - t_1$$

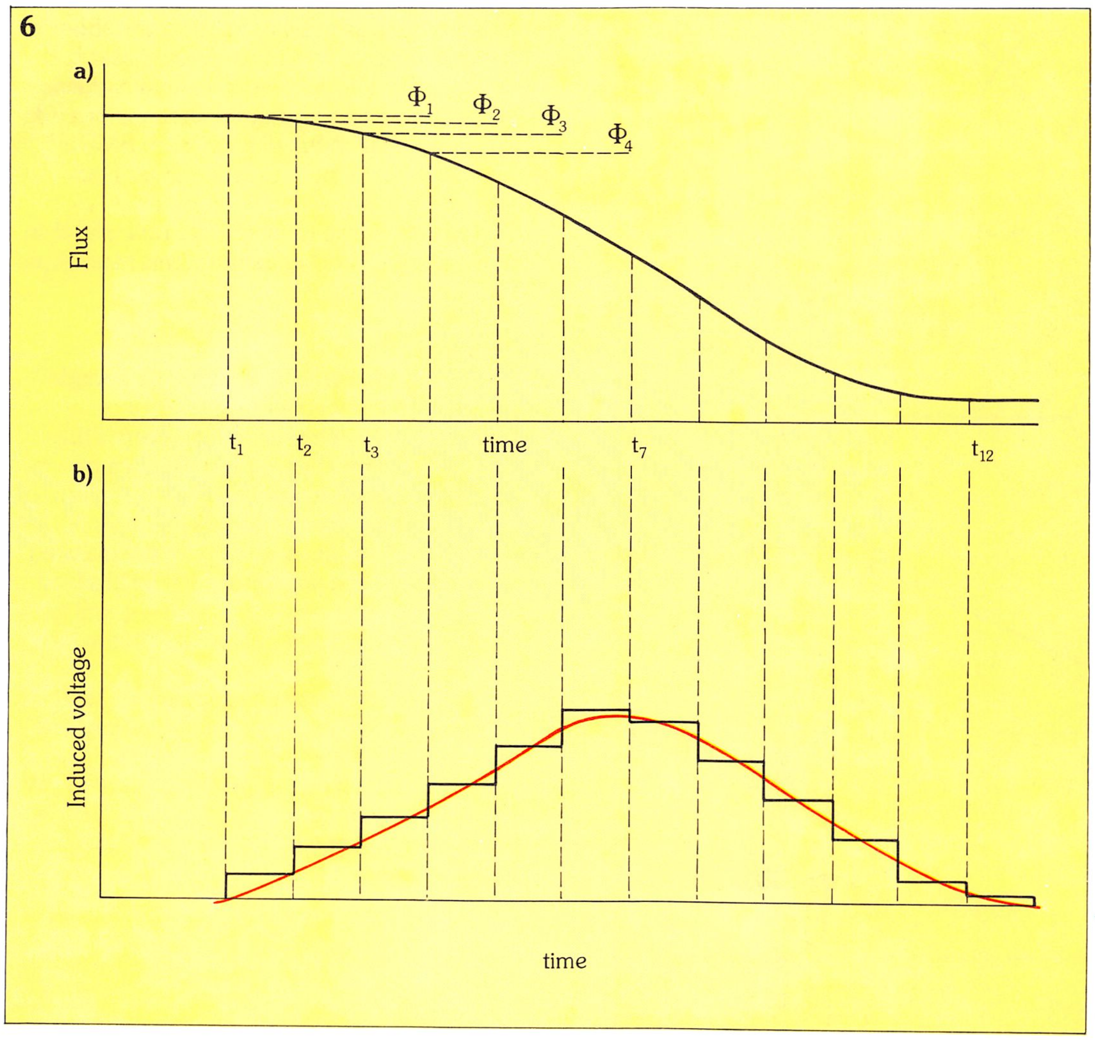
so the rate of change of flux linked, which is

5



5. A resistor replaces the voltmeter of figure 4a.

6



6. (a) Rate of change of flux with time; (b) rate of change of induced voltage with time.

the magnitude of the EMF is given by:

$$\frac{\phi_2 - \phi_1}{t_2 - t_1}$$

and it exists only during the period of time from t_1 to t_2.

The polarity of the voltage is another matter, however. If we replace the voltmeter in *figure 4a* by a resistor as shown in *figure 5*, then the induced EMF will obviously drive a current around the circuit. Now, most systems in a stable state tend to oppose any change, and we'll assume this is true of our system, so the current flow around the secondary circuit is shown by the direction of the current arrow. This current will create a magnetic field, H, in the direction shown, setting up a flux density in the same direction. This, as anticipated, will tend to oppose the original flux increase.

All of this is formalised by **Lenz's law**, which states that the direction of an induced EMF is such that it tends to set up a current opposing the change of flux (or the motion) responsible. You can see that the rule will also apply to situations where the induced voltage is caused by the motion of a conductor in a constant magnetic flux. This means that the induced EMF is actually equal, but opposite to, the magnitude of the rate of change of flux, i.e:

$$V = -\frac{\phi_2 - \phi_1}{t_2 - t_1}$$

As long as the rate of change of linked flux is constant, this is a perfectly acceptable model which gives us the value of the induced EMF. However, what happens if the rate of change of flux varies? Let's look again at the circuit in *figure 4a*, and say, for example, that the flux linked with the secondary changes in the manner shown in *figure 6a*. It is stationary before time t_1, but then starts to reduce, builds up to its maximum rate of reduction at time t_7, then slows down to a stationary value at about time t_{12}.

Now, at each successive instant of time (t_1, t_2, t_3 ...), the flux is ϕ_1, ϕ_2, ϕ_3 ... etc. So, during the interval t_1 to t_2, the flux falls from ϕ_1 to ϕ_2 and the induced EMF is given by:

$$V_1 = -\frac{\phi_2 - \phi_1}{t_2 - t_1}$$

Similarly, in the interval t_2 to t_3:

$$V_2 = -\frac{\phi_3 - \phi_2}{t_3 - t_2}$$

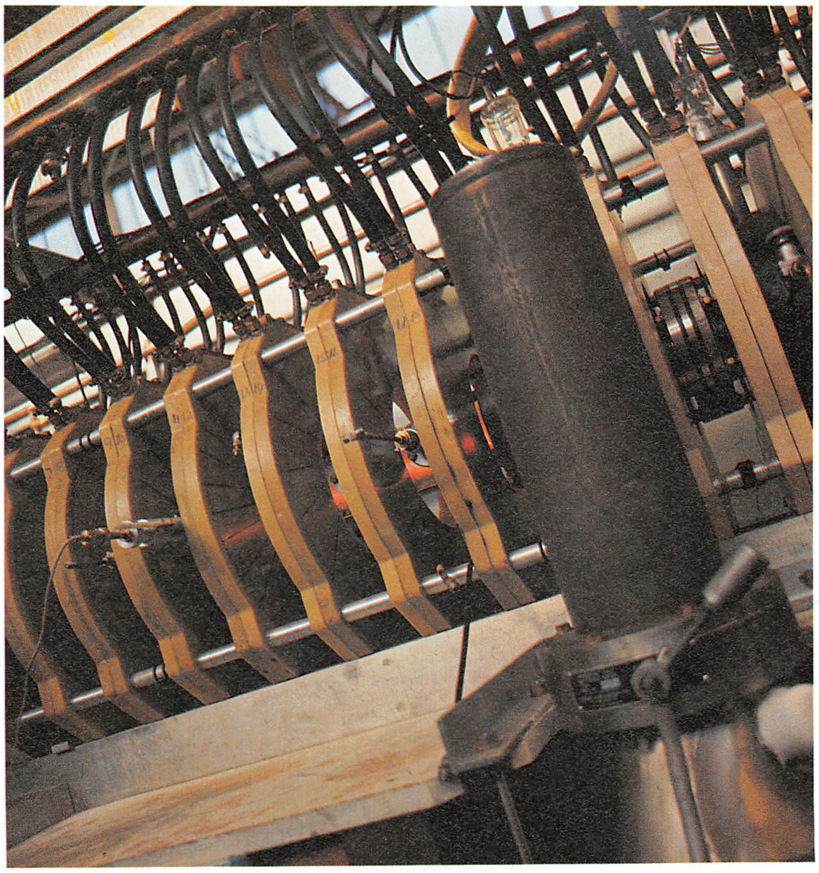
and so on. These individual EMFs are shown by the black, stepped line in *figure 6b*. If the intervals of time are reduced in length, the steps in the voltage graph will get smaller and smaller until the induced EMF becomes indistinguishable from the red curve. The induced curve starts at zero (when the flux is constant), increases to a maximum (when the flux is changing at its maximum rate) and then falls again to zero.

The Weber

Finally, we can use this concept of induced voltage to give the more usual definition of the unit of flux – the weber: a voltage of 1 V is induced in a circuit having a single turn of wire when the flux linked with it changes at a rate of 1 Wb s^{-1}.

Sometimes, as a result of this definition, the unit of 1 Vs is used, instead of 1 Wb. □

Below: neon plasma containment by superconducting magnets in a quartz envelope. This example is at the U.K. Atomic Energy Authority's laboratory at Culham.



Paul Brierley

11

DIGITAL ELECTRONICS

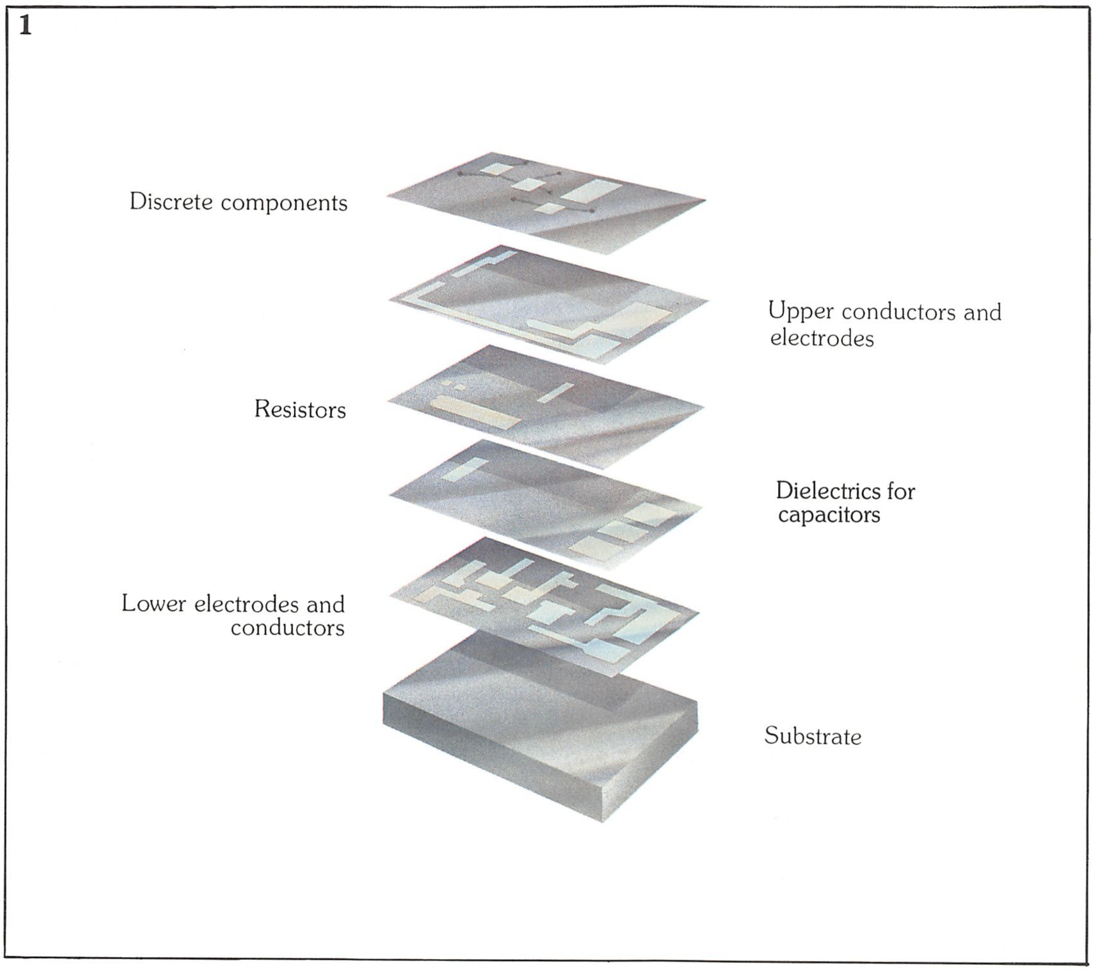
Hybrid integrated circuits

Thick and thin-film hybrid ICs

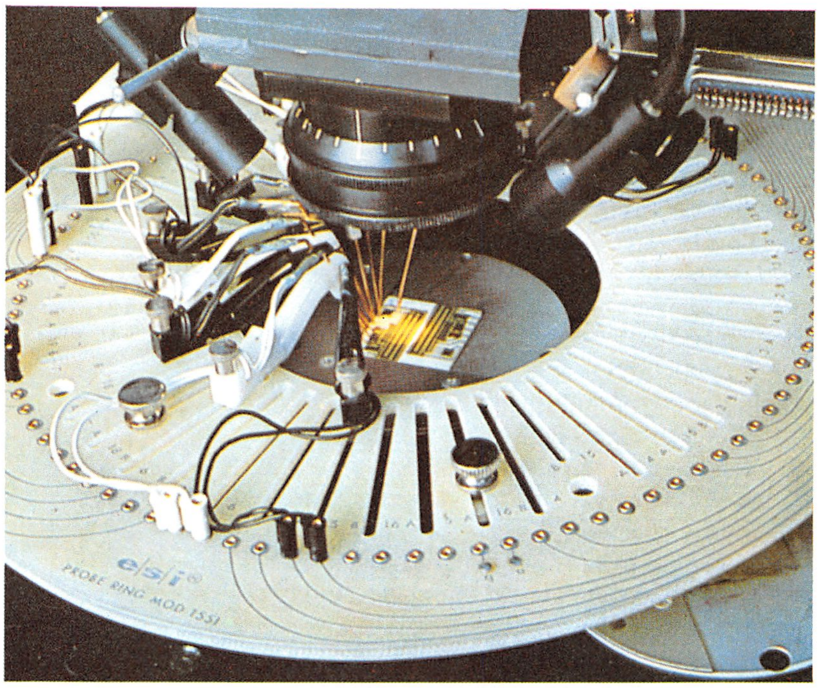
In *Digital Electronics 10* we examined the processes involved in the manufacture of integrated circuits. These, and all other ICs studied so far, are **monolithic** integrated circuits, i.e. all the components are fabricated on a single substrate. There is, however, a second type of IC – the **hybrid** integrated circuit.

Hybrid integrated circuits are essentially miniature electrical circuits. Resistors and capacitors are fabricated onto a substrate, alumina or beryllium oxide for example, and the active components – diodes, transistors and monolithic ICs – are then mounted on top, forming a layered structure (see *figure 1*). The individual components are connected either by thin gold wires or by a pattern of metallization. The whole assembly is then hermetically sealed.

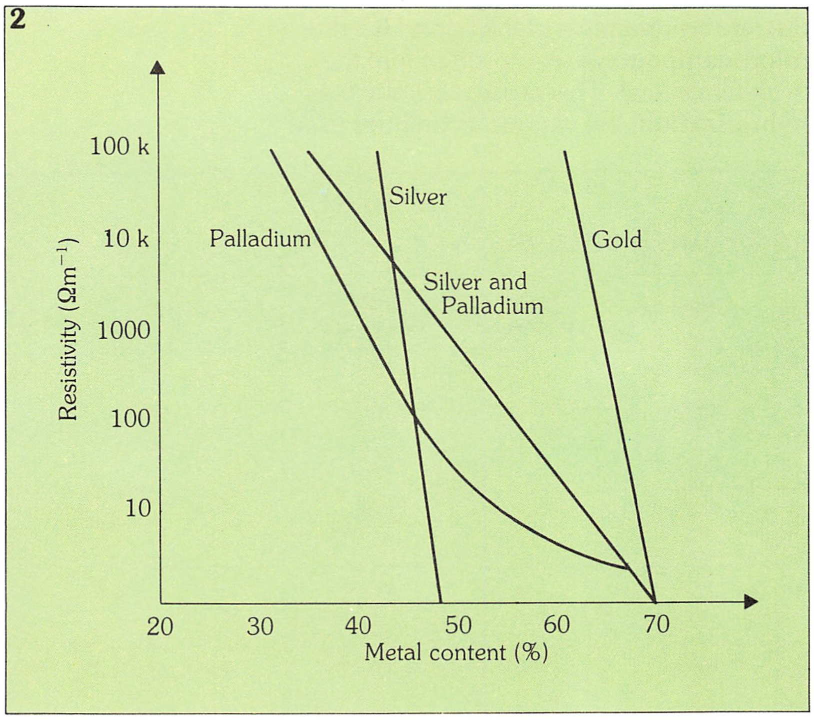
Most of the ICs in use today are monolithic – they were developed first, primarily because of the greater possibilities of miniaturization, a process vital to the growth of the electronics industry. However, as monolithic circuit technology


1
Discrete components
Upper conductors and electrodes
Resistors
Dielectrics for capacitors
Lower electrodes and conductors
Substrate

1. The layered structure of a hybrid integrated circuit.



2



Above: calibrating resistors on a thick-film hybrid IC using computer controlled lasers. (Photo: Neohm)

2. Resistivity *vs* metal content: note how a small change in the metal content significantly affects the resistance of the pastes.

reached maturity, it was realised that this type of IC had certain disadvantages and attention turned to hybrid circuits.

There are two types of hybrid ICs, thick-film and thin-film, so named because of the two different processes employed to make them.

Thin-film circuit elements can be obtained in two ways: either by vacuum evaporation or cathode sputtering. Vacuum evaporation deposits a layer of material – usually a metal – on the device's substrate. Cathode sputtering achieves the same result by the disintegration of a cathode (made of the material required) from a gas-discharge tube. The various elements can then be formed by photo-masking and etching processes.

As thick-film techniques are more popular, we shall now go on to discuss them in greater detail.

Making thick-film hybrid ICs

Thick-film circuits are made by a process of screen printing layers of dielectric, resistive and conductive pastes onto a substrate, which is then baked.

A 96% alumina mixture bound with silicates is usually used as the substrate as this provides the optimum mechanical, thermal and electrical characteristics necessary. Substrates can be bought from specialist manufacturers and are available in various standard sizes, the most widely used being a 5 × 5 cm square. The largest substrates, used for special applications, are 10 × 15 cm – larger substrates tend to warp when the ceramic material is baked and their electrical characteristics become difficult to control.

Vitreous dust and a conductive material are mixed together in an organic solution to make the component pastes suitable for printing onto the substrate. These pastes must have good properties of adherence; must not spread; and must allow overprinting when dry. The metals used for the conductive zones include gold, platinum, palladium and silver; whilst for the resistive zones platinum, ruthenium oxide, thallium oxide or silver palladium are used. *Table 1* sums up some of their characteristics. *Figure 2* illustrates how even a small change in the metal content of one of these pastes causes a large variation in its resistance. Ruthenium oxide is the most widely used metal in resistive pastes because it is the most stable compound.

Thick-film circuit manufacture begins with the layout design which is developed from the circuit diagram and is normally drawn ten to twenty times larger than actual size. The various construction stages correspond to the usage of the different pastes, conductive, resistive, etc., which are shown as different colours on the layout. Mylar film masters are then made for each colour layer which are then photographically reduced, to obtain an actual size photomask.

The circuit layout may be produced with the aid of a computer: the components being drawn on a VDU screen with a light pen; the computer then refines the design by finding the most economical and effective positioning for the circuit elements. A photoplotter – another digital machine – then automatically produces the photomasks.

Once the photomasks have been prepared, the image is transferred to the printing screen. Screen printing is a process that forces ink through a fine mesh; each layer of paste to be printed corresponds to a separate screen. The non-print areas of each screen are masked by a photographic process; the screen is then covered with photosensitive resins and exposed to ultraviolet light via the photomask. The unexposed areas are then washed away with a solvent to reveal the finished design.

The printing process used to make thick-films is strictly controlled – factors such as border definition, print repeatability and constant thickness of the deposited paste, are continually monitored. Each thick-film layer deposited on the substrate is left for five to ten minutes allowing capilliary forces to remove any imperfections left by the printing screen. The film is then dried at a high temperature (around 150 °C) to remove the solvents in the paste without affecting the organic elements binding it together.

The next stage in the production process is **sintering** – one of the most critical stages. Sintering removes the organic compounds from the printed thick film paste and fuses the silicon constituents that bind the paste to the substrate. Sintering is carried out at temperatures below the melting point of the paste constituents. The ovens used are automatically controlled within tight temperature limits.

Neither the processes nor the materials used in the manufacture of thick-film hybrid ICs allow very close tolerances to be obtained. Adjustments, therefore, have to be made after the sintering stage is complete: these will be covered in detail a little later in the chapter.

After any necessary adjustments have been made to the film components, the discrete components of the hybrid IC, the **microcomponents**, are inserted into the thick-film circuit. This process is known as **hybridization**; the various techniques used depending on the type of microcomponent.

If encapsulated components are to be connected, then point-to-point wiring, dip-soldering or solder reflow techniques are used. On the other hand, if non-capsulated, or naked components, are used then sophisticated insertion equipment is necessary to automatically connect monolithic devices. Two techniques are utilised: ultra-

Table 1
Characteristics of materials used for conductive zones

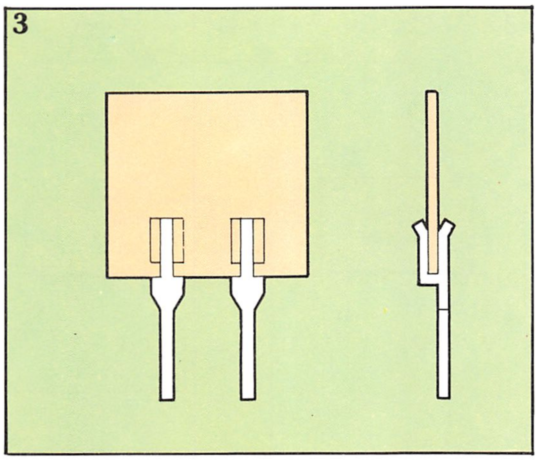
Composition	Cost	Conductance	Adhesion	Soldering resistance	Contact resistance	Migration
Gold	**	***	**	*	*	****
Platinum and Gold	*	*	***	*****	*****	*****
Gold and Palladium	***	**	***	****	***	****
Silver and Palladium	****	****	****	***	****	***
Silver	*****	*****	*****	*	**	*

sonic bonding and wire bonding.

Ultrasonic bonding fixes the chip to the substrate by an ultrasonically vibrating bonding probe. This generates localized heat which welds the component to the circuit film. Wire bonding uses gold wires to connect the chip to the thick-film circuit.

When all the component parts have been added, connecting pins are soldered onto the contact points on the edge of the substrate (see *figure 3*). The finished assembly is then encapsulated and can take the form of a DIL package or a special flat pack mounting (*figure 4*).

3



3. Connecting pins are soldered onto the contact points on the edge the substrate.

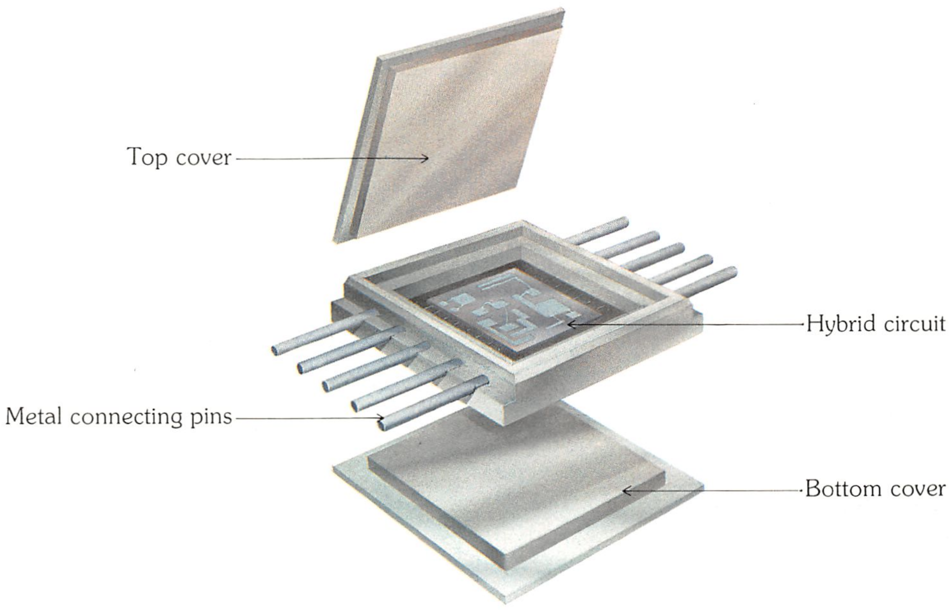
Adjustment methods

As we have mentioned, it is necessary to make adjustments to the values of the constituent components due to the wide tolerance range of materials and processes used. Thick-film resistors have a wide tolerance – typically around 15% – which has to be improved, usually to around 1%. This can be achieved by altering the thickness or shape of the resistor using electrolytic methods or ultrasonics respectively. Abrasives or lasers can also be used to alter shape. *Table 2* sums up the various methods.

Electrolytic adjustment

An electrolytic adjustment system is shown in *figure 5*. The resistor to be calibrated is covered with an electrolyte and connected to a control system. A probe is immersed in the electrolyte, and another is connected to one of the resistor's terminals. Both of these electrodes are connected to a DC power source. If the resistor is positively polarized, oxygen is liberated. This oxidizes the resistor's surface and increases its resistance. On the other hand, if the resistor is negatively polarized, hydrogen is

4



4. Flat pack mounting for the finished hybrid IC.

Table 2
Adjustment methods

Method	Technique	Comment
Electrolytic	Oxidation and reduction	Unsuitable for complex circuits
Ultrasonic	Vibrating probe	Economical
Abrasive	'Sand blasting'	Economical
Laser	Intensive heat	Fast, accurate and expensive

liberated thus chemically reducing the resistor and lowering its value. In either case, the process is stopped when the control system indicates that the correct resistance value has been reached (see *figure 6*).

Although very accurate resistances can be obtained with this method, it is unsuitable for complex circuits, as the resistor under adjustment must be electrically isolated from other circuit elements.

Ultrasonic adjustment

This method calibrates thick-films by removing resistor material with an ultrasonically vibrating probe. This is a modified version of the bonding probe mentioned earlier. By moving the probe along the length of the resistor, material can be removed until the desired value is reached. A constant flow of nitrogen is directed onto the substrate during this process, to remove the resulting debris.

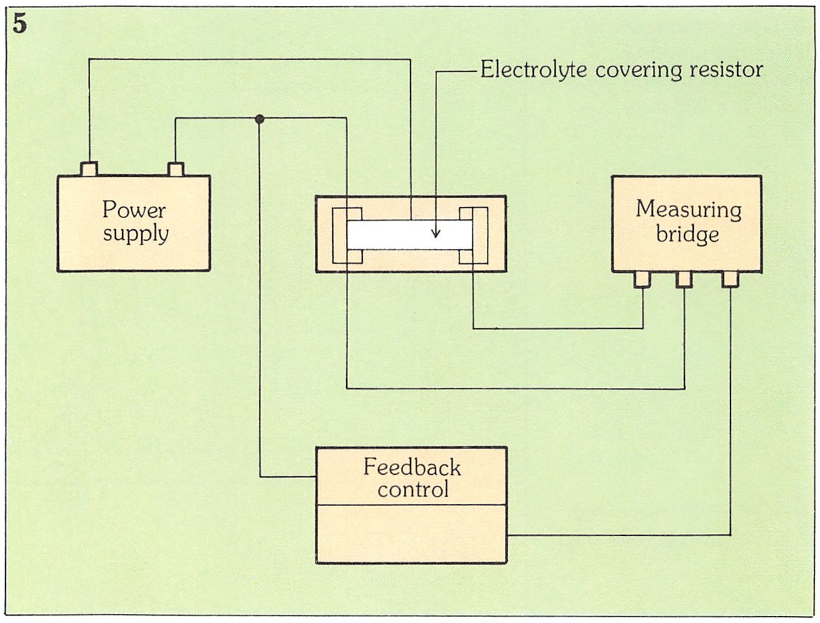
Abrasive adjustment

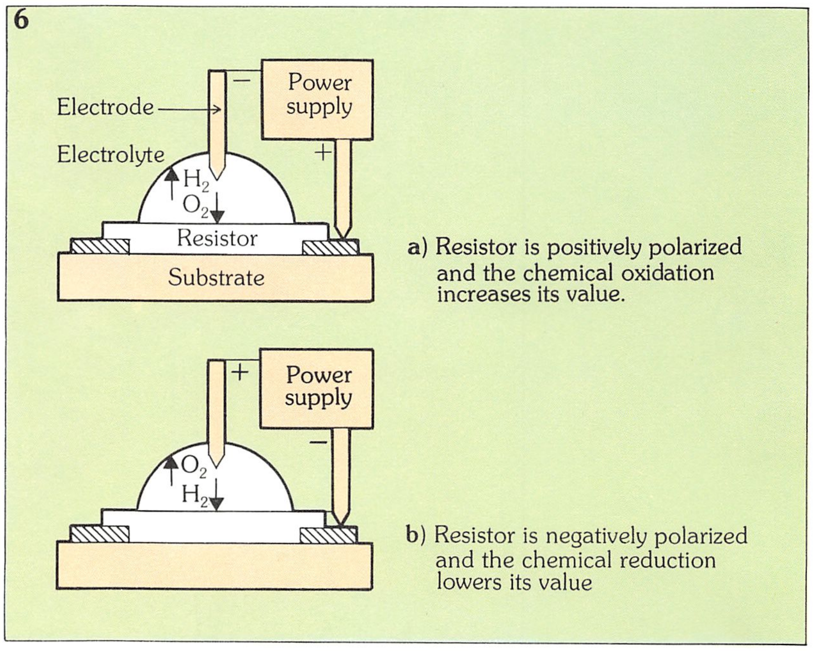
The abrasive adjustment process is illustrated in *figure 7*. Quite simply, the surface of the resistor is eroded by a high pressure jet of air mixed with abrasive particles – like a miniature sandblaster. The resistor's size is reduced causing its value to be lowered. As in the other adjustment processes the resistor is connected to a high precision measuring bridge, and the abrasion is stopped when the desired value is reached.

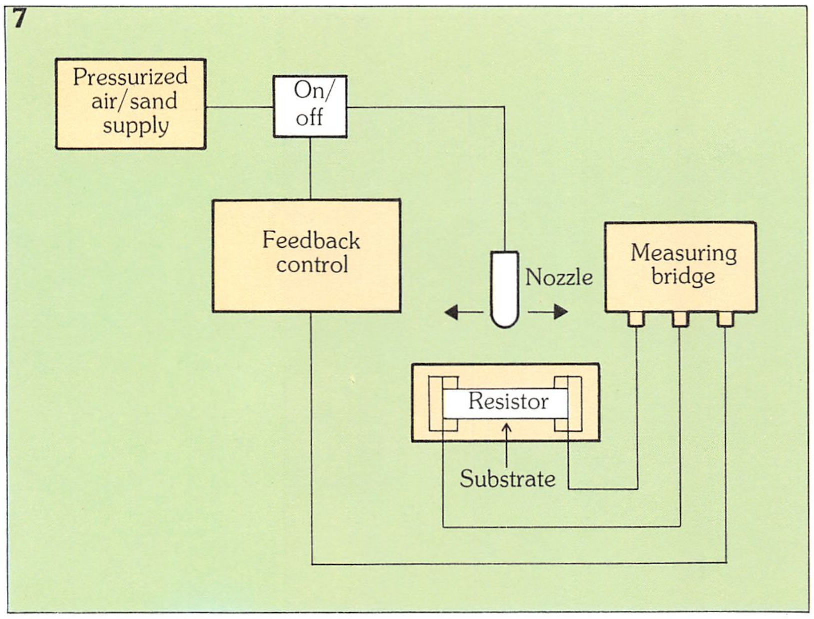
Although there are many advantages to this system: high speed; low temperature operation; immunity to the danger of cracking the thick-film or substrate; and, above all, a very high level of accuracy,

5. An electrolytic adjustment system.

6. Calibrating the resistor.

5


6

Electrode
Electrolyte
Power supply
−
+
H_2
O_2
Resistor
Substrate
a) Resistor is positively polarized and the chemical oxidation increases its value.
Power supply
+
−
O_2
H_2
b) Resistor is negatively polarized and the chemical reduction lowers its value

7


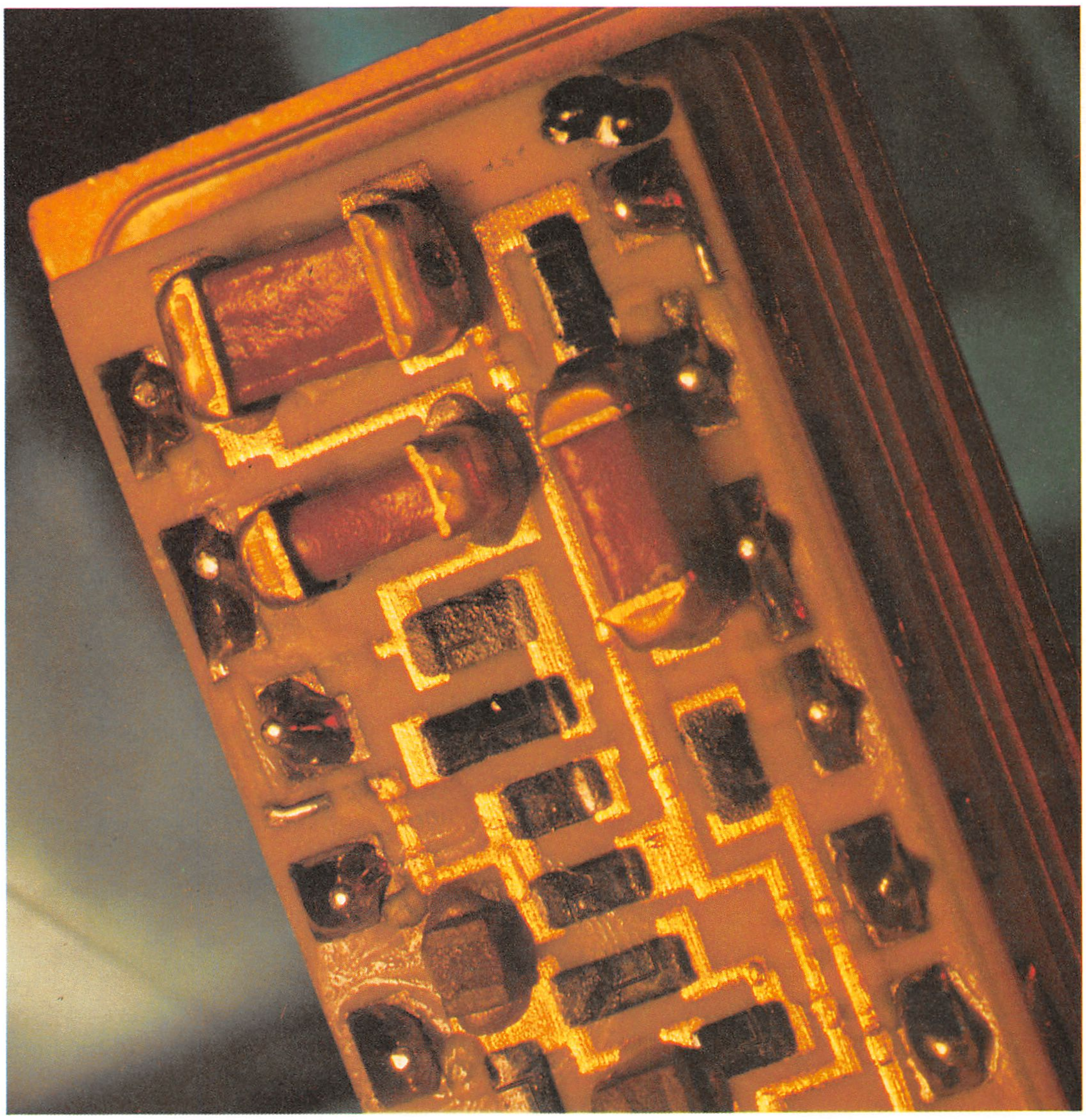
there are disadvantages. These include: the high level of electrical and thermally derived noise; the unsuitability for small surface areas; the abrasive dust gets everywhere; and, most importantly, the circuit is not hermetically sound.

Laser adjustment

This is the latest technique, and uses a finely focused laser with either a continuous or an impulse beam. The laser, as in the abrasive process, is used to cut away parts of the resistor, to lower its resistance. The cuts made are generally straight, parallel, or serpentine; other types of cut can also be made. Some examples are shown in *figure 8*.

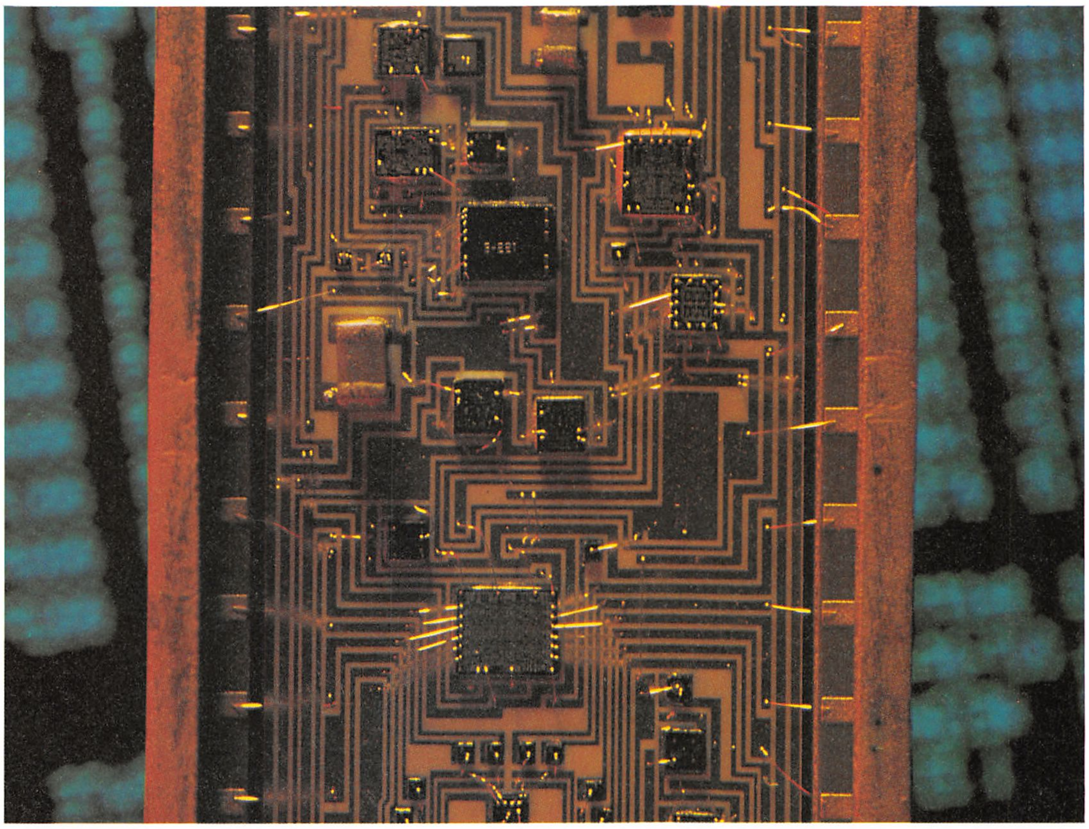
Laser adjustment has many practical advantages:

7. The abrasive adjustment process.



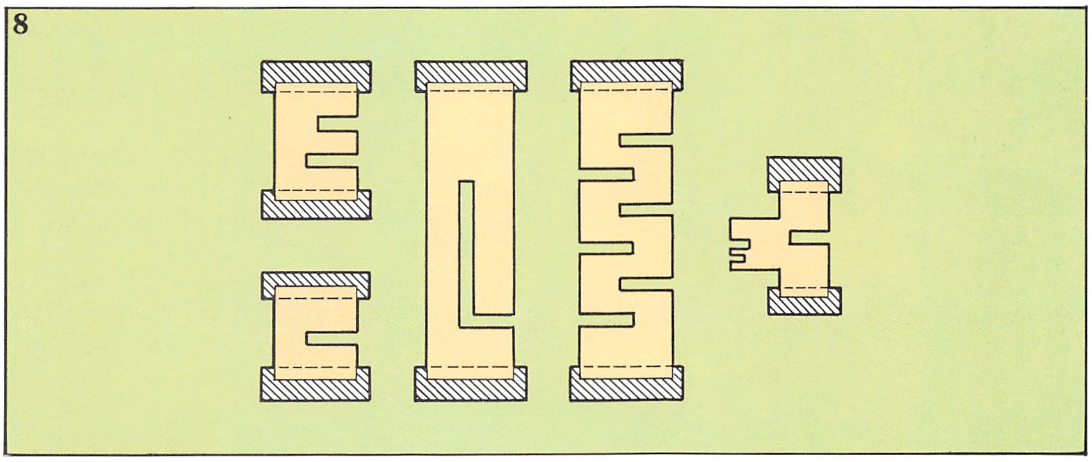
Right: a thick-film hybrid IC after assembly of the passive components.

Paul Brierley


Paul Brierley

Left: **thin-film hybrid IC** – seen here set against the display of a VDU.

8


8. Examples of the types of cut possible during laser adjustment.

1) the heat of the laser hermetically seals the cut – preventing any moisture from entering the film;
2) it is very fast – cuts can be made at the rate of one to two centimetres per second, and an entire circuit can be adjusted in one pass. Over 15,000 resistors can be calibrated in an hour;
3) resistors with a very small area – down to 1 mm^2 – can be adjusted;
4) very close tolerances can be reached – as low as ± 0.5%.
The main disadvantage is cost: equipment and maintenance costs are high.

The problems of adjustment

Although adjustment is necessary to 'fine tune' the characteristics of the thick-film resistor, other properties are also changed: temperature stability and noise characteristics suffer most. Manufacturers, needing to guarantee the quality of their circuits, can precisely evaluate these variations and compensations can therefore be made for them in the processes that precede the adjustments.

Hybrid and monolithic ICs compared

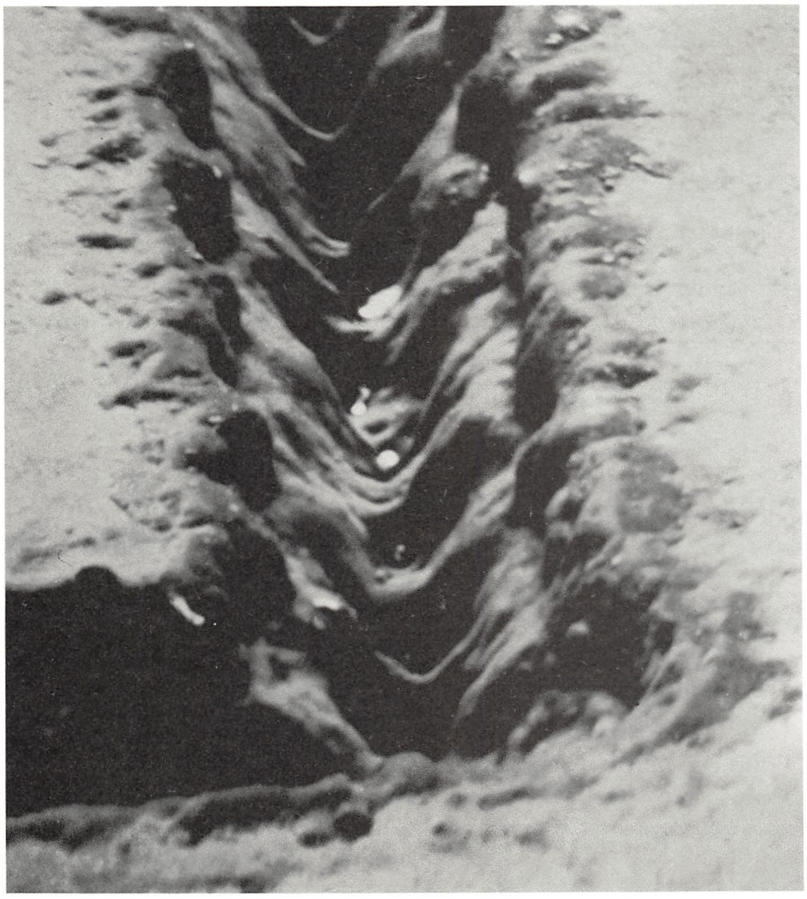
Thick-film technology is very versatile: resistors and capacitors can be made in a wide variety of combinations and values, and many different discrete components (active semiconductors or passive components) can be fixed onto thick-films, thereby creating a wide range of possible ICs.

Thick-film circuits are often used for medium to high power and high voltage applications. Very close tolerance levels can be achieved, and some of these circuits can work at very high frequencies. Thin-film devices are preferred in applications that demand high stability and reliability.

It can generally be said that thick-films are used where: thin-film hybrids and monolithics are uneconomical; where the circuit requires power, voltage and frequency levels, or temperature coefficients and tolerances unobtainable from monolithics; where assembly times and circuit size need to be reduced; or where improved reliability in respect of discrete circuits is needed. Most thick-film hybrid circuits are designed to solve special application problems, but this technology also lends itself well to standard volume production.

To conclude, let's look at the advantages offered by hybrid technology. These can be grouped under the headings: performance, flexibility, reliability and economics.

Below: an L-shaped cut made during laser adjustment.



Performance

1) A traditional prototype circuit using discrete components can be converted to a hybrid circuit without changing the circuit design, with a subsequent saving in space.
2) Both thick and thin-film circuits can be reproduced in high volume quantities without affecting their performance.
3) The high thermal conductance of the substrate material limits the thermal difference between components. This reduces any thermally derived variations in performance.
4) Resistors can be adjusted under operating conditions, and it is thus possible to compensate for variations in all the passive components.
5) The resistors can dissipate high power levels.
6) Very close resistance tolerances can be obtained. Low cost capacitors can be made, using the substrate as the dielectric.
7) Manufacturing techniques allow high dielectric isolation to be obtained. This is a distinct advantage when the circuit has to work at high voltage or frequency levels.

Flexibility

1) Specific requirements can be easily met by designing the circuit in conjunction with the user.
2) Modifications to the original design can be easily made at low cost.
3) Small variations in the finished product can be easily obtained with small changes in the production process.
4) A prototype circuit can be quickly transferred to production.

Reliability

1) Many tests have demonstrated the high reliability of hybrid circuits – especially those using the thin-film technology.

2) The high level of reliability is essentially due to the low number of interconnections between the various components – something that is impossible to obtain with discrete components.
3) The reduction in the number of interconnections also reduces the defects associated with soldered joints, e.g. loss of contact through mechanical shock and vibration and parasitic resistances caused by imperfect electrical contacts.
4) Given that the deposited material and the substrate merge together, and that the substrate has a good thermal conductivity, a build up of heat in localised areas is minimized.

Economics
1) Production costs are low, since most of the work can be automated and calibration can be carried out on whole circuits at once.
2) Materials and equipment are readily available.
3) Investment in equipment and staff training is relatively low.
4) Production line controls, visual inspection and assembly procedures are much reduced, compared to other fabrication technologies.

Glossary

adjustment	processes that correct the values of thick-film resistors. Can use electrolytic, ultrasonic, abrasive or laser techniques
alumina	aluminium oxide. Used to make hybrid circuit substrates
bonding	processes that fix discrete components and connecting pins to circuits. Using soldering, wire bonding or ultrasonic methods
hybridization	techniques used to make a hybrid integrated circuit – that is one that uses a combination of integrated, discrete and printed components. Hybrid circuits can easily replace ordinary circuits, with a consequent saving in space and a gain in reliability
mylar	polyester sheet used to make photographic films and masks
photoplotter	digitally controlled machine that automatically produces the photomasks used in integrated and hybrid circuit manufacture
sintering	heat process that removes the organic compounds from printed thick-film paste, and fuses the silicon constituents which bind the film to the substrate
screen printing	method used to print the connections and resistive layers of thick-film paste onto a substrate
thick-film	the conductive and resistive elements of hybrid circuits can be made from thick-films. These are layers of pastes that are screen printed onto an alumina substrate and are between 12 and 20 μm deep
thin-film	hybrid circuits can also be made from thin-films. In this case the conductive and resistive elements are formed by vacuum evaporating or cathode sputtering. Thin-films rarely exceed a depth of 0.5 μm

BASIC COMPUTER SCIENCE

8

Programming

What is a program

In previous chapters we have seen how the various **hardware** components of a computer system relate to each other and how they work. This hardware, however, can do nothing by itself, it needs instructions to tell it what to do. These instructions, the **software**, must be given to the computer in a step-by-step sequence known as a **computer program**.

The concept of a program exists outside the computer environment, for example a cookery recipe and a car service manual are types of programs. However, they differ from the computer program in that they assume a degree of intelligence in the reader, i.e. each minor step, for example:

1) place one pint of water in a saucepan;
2) place saucepan (with water in) on cooker;
3) turn on cooker;
4) wait till water boils;
5) take egg from fridge;
6) place egg in saucepan;

Below: programs can be written for almost any office environment – here we see one at work in an insurance office.

Tony Stone Associates

7) wait three minutes;
does not need to be detailed, only each main step, i.e:
1) boil an egg for three minutes.

A computer, however, is *not* intelligent – it simply follows the instructions that have been programmed into it, in other words, a computer is only as good as the software that has been written for it. These programs, then, must be designed and written with the required task in mind to control the computer through *each step*.

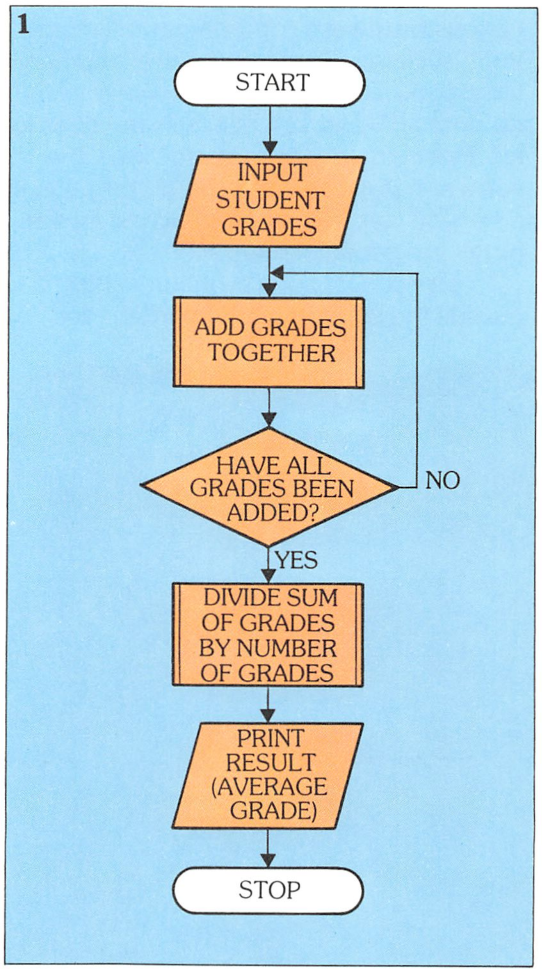
Although a program can be designed and written in many ways to accomplish the same end result, some methods have definite advantages over others. For example, one may use more instructions, require more computer memory space, and take more time to **run** (i.e. execute) than another.

A common misconception amongst beginners is that the task of programming is simply the writing of a list of instructions in one of the so-called **high-level languages**, such as BASIC, FORTRAN or Pascal. In fact, this forms only one of the six (equally important) separate stages in the writing of a program. These are:
1) understanding the problem;
2) designing the program;
3) writing the program instructions;
4) translating the program;
5) testing;
6) documentation.

Understanding the problem
This first step in program development is perhaps the most important. If a programmer jumps straight into the design stage without a complete understanding of the problem to be solved then, almost inevitably, there will be problems. A clear and complete description or **specification** of all possible **inputs** and **outputs**, as well as all the **processes** involved, is needed so that a comprehensive and correct solution to the problem is attained. The form of this specification may vary from a written narrative description, to a list of statements, a diagram, or a combination of all three. Tables may also be used to specify conditions.

Let's examine a simple problem with the following specification: a teacher wants a computer program to help assess a class of students. The program is to determine each student's final grade and produce it as the program *output*. Six previous grades are program *inputs*, and the main *process* involved in the program is to calculate the average of all six grades. This average result is the final grade.

We can help analyse this specification by writing a list of what the program has to do:
1) input the six individual grades;

1

START
INPUT STUDENT GRADES
ADD GRADES TOGETHER
HAVE ALL GRADES BEEN ADDED?
NO
YES
DIVIDE SUM OF GRADES BY NUMBER OF GRADES
PRINT RESULT (AVERAGE GRADE)
STOP

1. A list drawn up from the specification provided for a computer program to assess students' grades.

2) calculate the average of the six grades;
3) print out the average grade.
The list can be drawn as shown in *figure 1*.

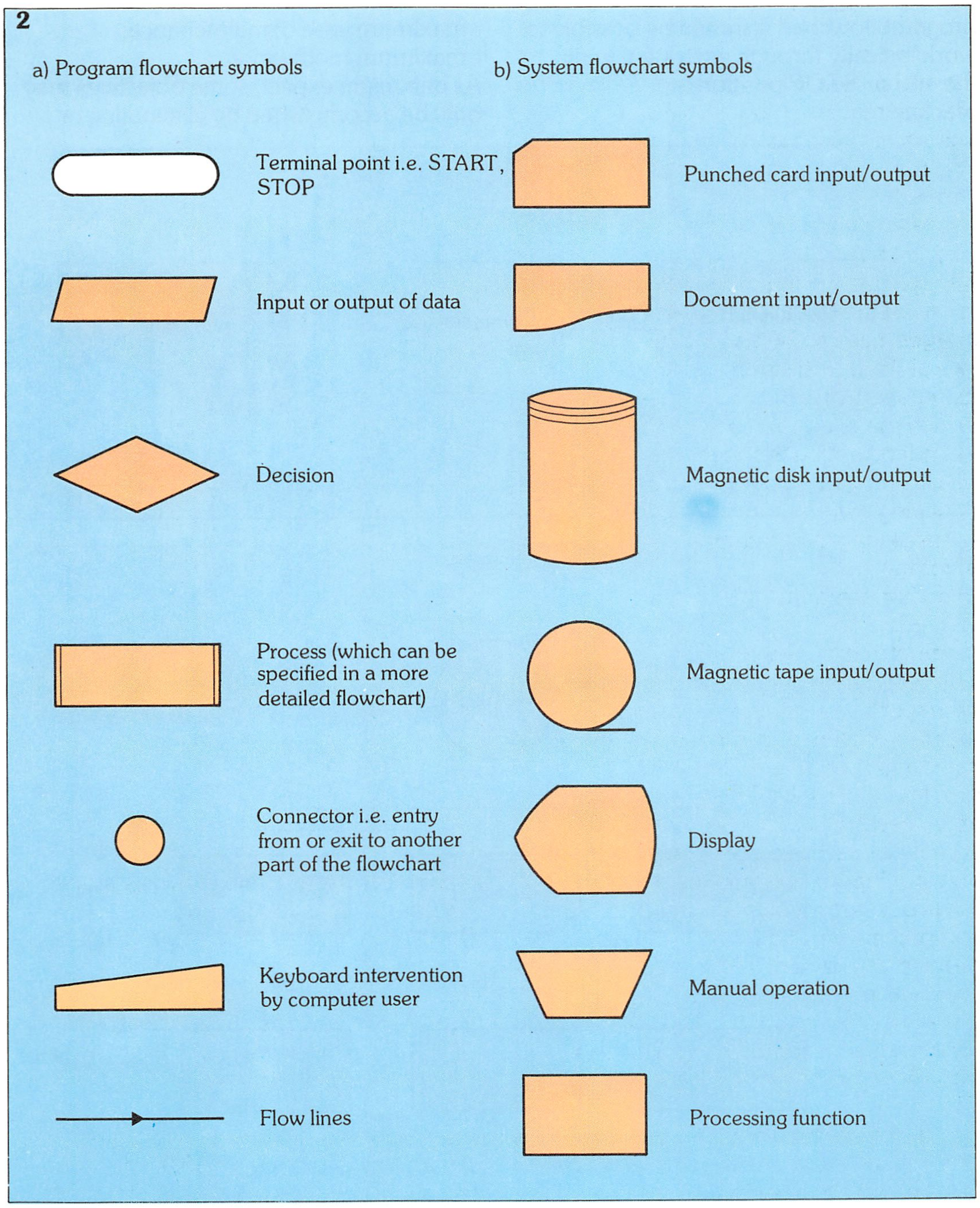
Flowcharts
Figure 1 is an example of a **flowchart** – it is a graphical representation of the processes required to achieve the end result. If drawn correctly, flowcharts present the structure of a program so that the relationship between stages can be easily understood. This particular type of flowchart is known

as a **program flowchart** (because it represents details of the actual computer program) and, because the program isn't shown in any great detail, it is an example of an **outline program flowchart**. Later, when the program is designed and formal statements in a programming language are used, a **detailed program flowchart** is drawn.

The outline symbols used in the flowchart of *figure 1* are redrawn in *figure 2a* with definitions; other common program flowchart symbols are also shown. Flow direction in a flowchart is generally from top to bottom and from left to right, but arrowheads are used to indicate direction preventing any doubt.

Another type of flowchart often used when developing computer programs is the **system flowchart**. This performs a similar function to the program flowchart showing how the separate parts of the system, e.g. disks, tapes, printers etc. are used by the program. *Figure 2b* shows a

2. (a) Program flowchart symbols; (b) system flowchart symbols.


2
a) Program flowchart symbols
b) System flowchart symbols
Terminal point i.e. START, STOP
Input or output of data
Decision
Process (which can be specified in a more detailed flowchart)
Connector i.e. entry from or exit to another part of the flowchart
Keyboard intervention by computer user
Flow lines
Punched card input/output
Document input/output
Magnetic disk input/output
Magnetic tape input/output
Display
Manual operation
Processing function

selection of common symbols used in the construction of system flow charts.

The system flowchart corresponding to our example of a computer program to produce average grades, is shown in *figure 3*. From this simple flowchart we can see that data is input to the computer from a document and, after processing, is output to a printer.

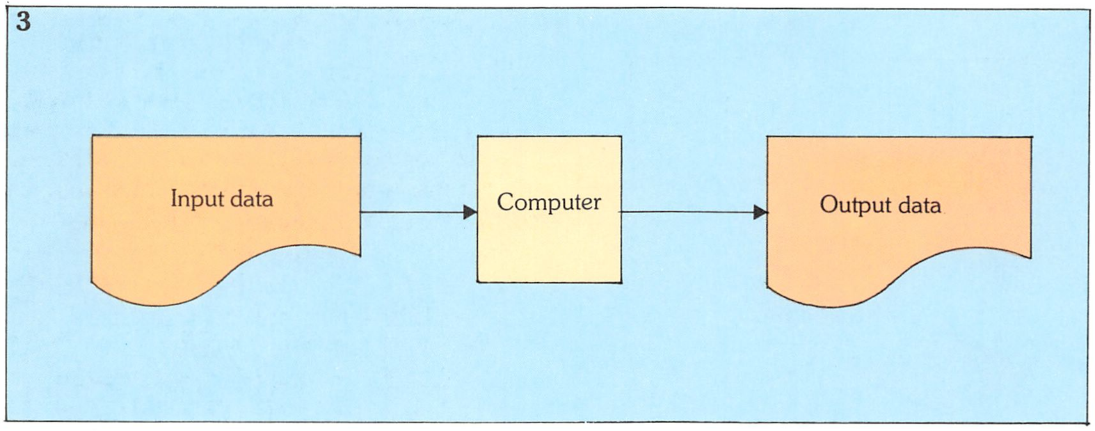
Properly produced flowcharts (program or system) show the sequence of steps in a program and the links between them. By beginning at the **START** position in the program flowchart it should be possible to work logically through each step, until the end or **STOP** position is reached.

Designing the program

As we have said, programs can be designed in different ways to achieve the same result. Objectives which may influence the design of a particular program include:
- minimum cost;
- minimum program size;
- minimum data storage size;
- minimum running time;
- maximum flexibility;
- maximum reliability;
- maximum ease of maintenance;
- maximum modularity.

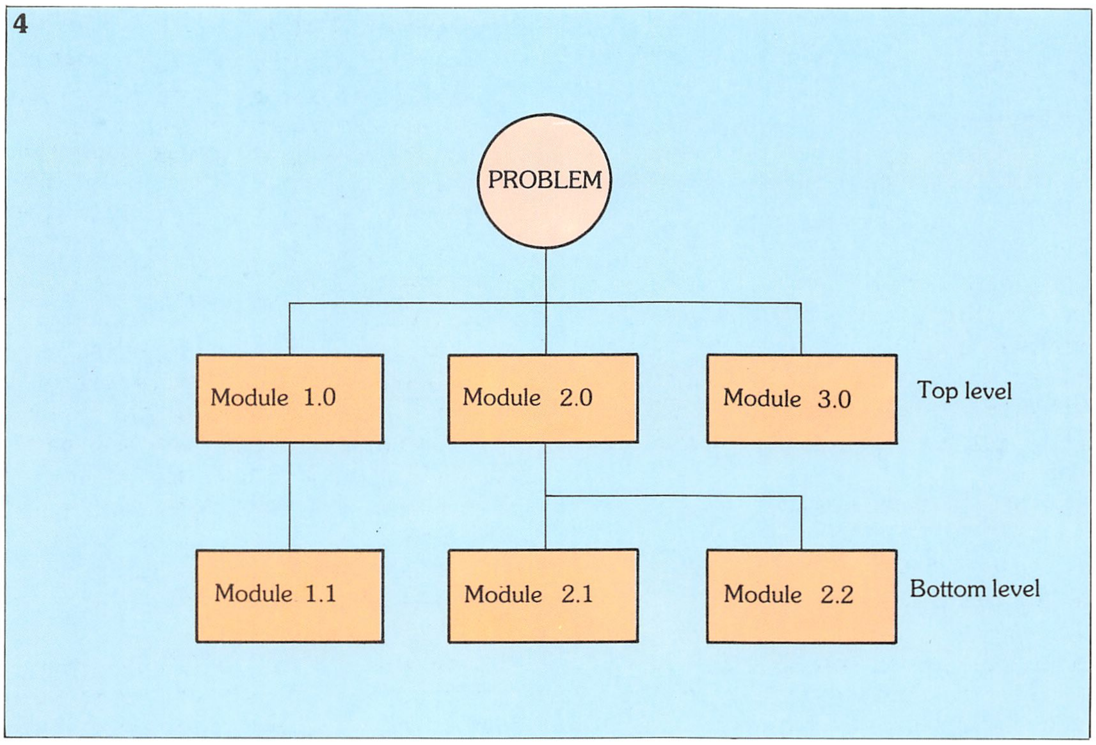
As one might expect, some objectives may only be accomplished by eliminating or

3



3. System flowchart for the program to produce average grades.

4



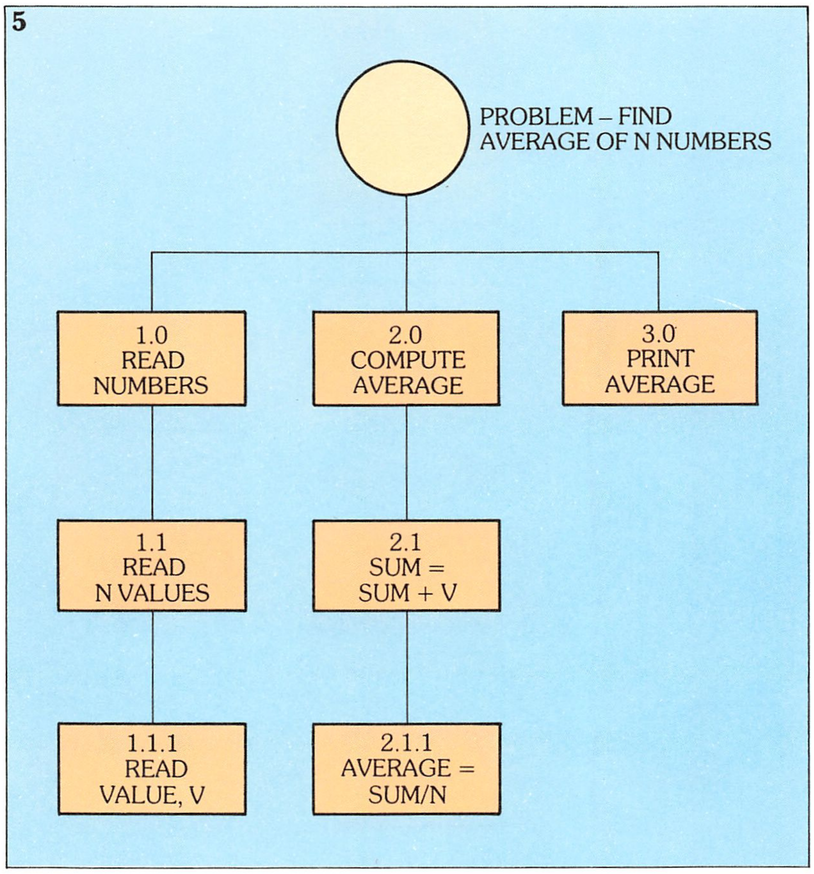
4. The top-down design process illustrated in block diagram form.

compromising others, and so the *final* program design will be the result of compromises between objectives, design and, with some applications, end result.

There are two fundamental approaches to program design: **bottom-up design** and **top-down design**. Bottom-up design involves the definition of all the *details* which are then built up to make the whole program. Although popular in the past, this method is not often used because of its inherent disadvantages. For example, when beginning to design a program in complex detail, it is all too easy to lose sight of the objective. Errors are easily made and problems arise when trying to interface, i.e. fit together, the disparate parts of the program.

5. Top-down design for the program to produce average grades.

5



The top-down design method is generally preferred because it helps to resolve the difficulties encountered in bottom-up design. Essentially, the concept of top-down design is a *systematic* procedure for problem solving which entails looking first at the *specification* of the solution to the problem, rather than looking at the solution itself. This specification is usually written in quite general terms and, in fact, forms the **top level** of the program.

Each statement within this top level specification is then broken down into its constituent parts to form the second level, which is itself broken down in a similar way, and so on until no more detail is necessary; the last level is known as the **bottom level**. This top-down design process is illustrated in block diagram form in *figure 4*. We can see how the problem is specified as a number of statements (modules 1.0, 2.0 and 3.0) forming the top level; these statements are then broken down where necessary to form the bottom level (modules 1.1. 2.1, 2.2). Each block in *figure 4* has been labelled a module, and should be structured independently of the others, to permit separate development and testing. These modules can be thought of as being **sub-programs**.

Reconsidering the earlier example of the teacher's program, the list which was compiled from the specification can be used as the top level of the top-down design shown in *figure 5*:
1) input the six individual grades, i.e. module 1.0, READ NUMBERS;
2) calculate the average of the six grades, i.e. module 2.0, COMPUTE AVERAGE;
3) print out the average grade, i.e. module 3.0, PRINT AVERAGE.

The second level is formed by modules 1.1, READ N VALUES and 2.1, SUM = SUM + V. These are program instructions relating to the input and addition of the grades.

The bottom level comprises modules 1.1.1, READ VALUE, V (relating to each individual grade) and 2.1.1, AVERAGE = SUM/N (i.e. the average grade is the total sum divided by the number of grades).

Although this is a fairly simple example, it does illustrate how a problem can be divided into more and more detailed levels to make the solution simpler. It is much easier to write sub-programs, and verify they are correct, for small modules than it is to write large complicated modules which include a number of different or unrelated operations.

The flowchart in *figure 6* shows the related module numbers beside each program stage and also illustrates how each module fits into a higher level. Modules of complex programs however, will not always be so easy to relate into higher levels. This flowchart is similar to that of *figure 1* but uses more concise statements, similar to those of some high-level programming languages.

Writing the program instructions

Program writing, often called **coding**, translates the program design into a language that can be understood and used by the computer. This language may be: machine code; a low-level programming language or assembly code; or a high-level language such as BASIC, COBOL or FORTRAN. The flowchart originally drawn in the design stage can be modified to suit whichever programming language is to be used, but regardless of language, the following fundamental instruction types are common to all:
1) data movement – used to transfer contents of memory locations. Examples are **move**, **load** and **store**;
2) arithmetic and logic – for performing addition, subtraction, AND, OR, NOT, etc. (multiplication and division are usually only available in high-level languages);
3) transfer of control – permits sub-programs and loops to take place within a program;
4) input and output – used to control the reading of data into the computer and the printing or displaying of data from the computer.

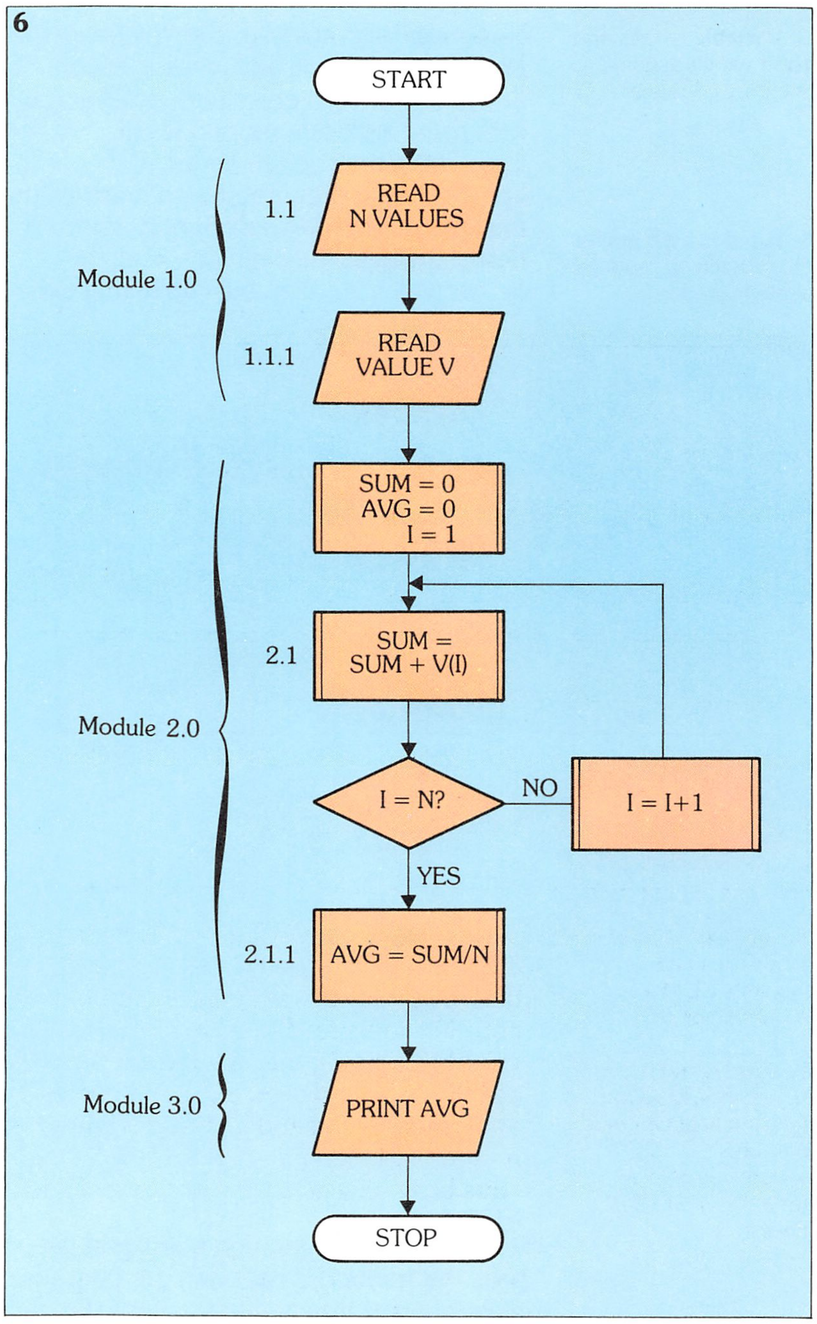
Generally, the higher the level of computer programming language used, the simpler these fundamental instructions are to use. However, this is not always the case as some high-level languages are designed with specific applications in mind, and are therefore best suited to these applications. For example, COBOL (**com**mon **b**usiness **o**riented **l**anguage) has simple business-type instructions and is therefore used most successfully in business programs. In a mathematical application, a language such as ALGOL (**algo**rithmic **l**anguage) would be easier to use because its simpler mathematical instructions would help the programmer.

Running the program

Suppose that the program has been loaded into the computer's memory, the computer has a terminal as an input peripheral, and data has been prepared for each student.

The first thing to be done is to instruct the computer to begin working on our problem, which we do with the START instruction. Following instructions (modules 1.1 and 1.1.1) tell the computer to read in the quantity of grades (N = 6), storing the quantity in a memory location named N, and then read in the individual grades, storing them in memory locations

6



6. Flowchart showing how module numbers are related to each program stage.

7. Variable names and their values assigned to memory addresses.

7

Memory address	001	002	003	004	005	006	007	008	009	010
Variable name	N	V_1	V_2	V_3	V_4	V_5	V_6	SUM	I	AVG
Initial contents	6	95	92	86	90	100	89	0	1	0
Final contents	6	95	92	86	90	100	89	552	6	92

8

	I = 1	I = 2	I = 3	I = 4	I = 5	I = 6
Accumulator	0	95	187	273	363	463
Variables V_1 to V_6	95	92	86	90	100	89
Variable SUM	95	187	273	363	463	552

9

	I = 1	I = 2	I = 3	I = 4	I = 5	I = 6
Accumulator	95	187	273	363	463	552
Variables V_1 to V_6	95	92	86	90	100	89
Variable SUM	0	0	0	0	0	552

8. How the computer leads the value of variable SUM into the accumulator.

9. With some languages, repeated loading of accumulator contents into the variable SUM location (as in figure 8) is not necessary.

V_1 to V_6 as shown in *figure 7*. Note that decimal numbers have been used, not binary, for our convenience. Memory locations N and V_1 to V_6 are known as **variables**, that is, they are locations which are allocated and used for the values associated with N and V_1 to V_6 in the program. Similarly, memory locations have been allocated for variables SUM, AVG, and I.

Figure 7 shows that the three variables SUM, AVG and I have initial contents of 0, 0 and 1. This is due to the module with the program statements:

```
SUM = 0
AVG = 0
  I = 1
```

This is known as **initialising variables** and allows the computer to allocate particular memory locations for each variable and set the contents of each location to the required value.

The SUM location is used to store the sum of the input grades and the AVG location stores the average of those grades. Variable I, however, is used to count the number of **iterations**, i.e. the number of times the addition sub-program (I = I + 1) is carried out, which identifies the successive values of V in memory. This is the reason why variable I, is initialised to 1 by the initialising module.

The addition module forms a **loop** in the program such that the addition, i.e:

```
SUM = SUM + V(I)
```

will be carried out N times, because of the decision statement:

```
I = N?
```

where N is set to the number 6, whereupon the loop is broken.

This process can be seen in *figure 8*, which shows how the computer leads the value of variable SUM into the accumulator. The value of the variables V_1 to V_6 are then successively added to the contents of the accumulator; after each addition the contents of the accumulator are loaded into the variable SUM location. With some computer programming languages this repeated loading of accumulator contents into the variable SUM location is not necessary and is only done when the final addition has been performed (shown in *figure 9*).

When variable I is less than N (where N = 6) then the decision statement I = N? produces a NO answer, so the addition loop is re-iterated. However, when I = 6, i.e. the loop has been performed six times, the decision statement produces a YES answer, so the loop is broken and the program moves to the next stage (module 2.1.1). The statement:

```
AVG = SUM/N
```

instructs the computer to divide the contents of variable SUM by the contents of variable N and store the contents in variable AVG.

Module 3.0 tells the computer to print variable AVG – the aim of this example program. Now, everything required has been accomplished, for one student's final grades, so the computer is instructed to stop working on our program with the instruction STOP.

There are two important points we should notice from this description of how the program runs:

1) The computer can make decisions. By making a simple statement such as:

I = N?

the computer decides whether this is true (YES) or false (NO) depending on the information with which it has been provided. The next step which the computer takes depends on the answer.

2) The program has flexibility, since the variable names N and V were used for the inputs. This allows the program to process any number of grades, not just six, and many different values for the grades.

Improvements

The flowchart of *figure 6* illustrates a program for determining the average grade for one student, but it must be repeated separately for each student in the class. We can alter the program to that shown by the flowchart in *figure 10* where a decison box has been added which asks:

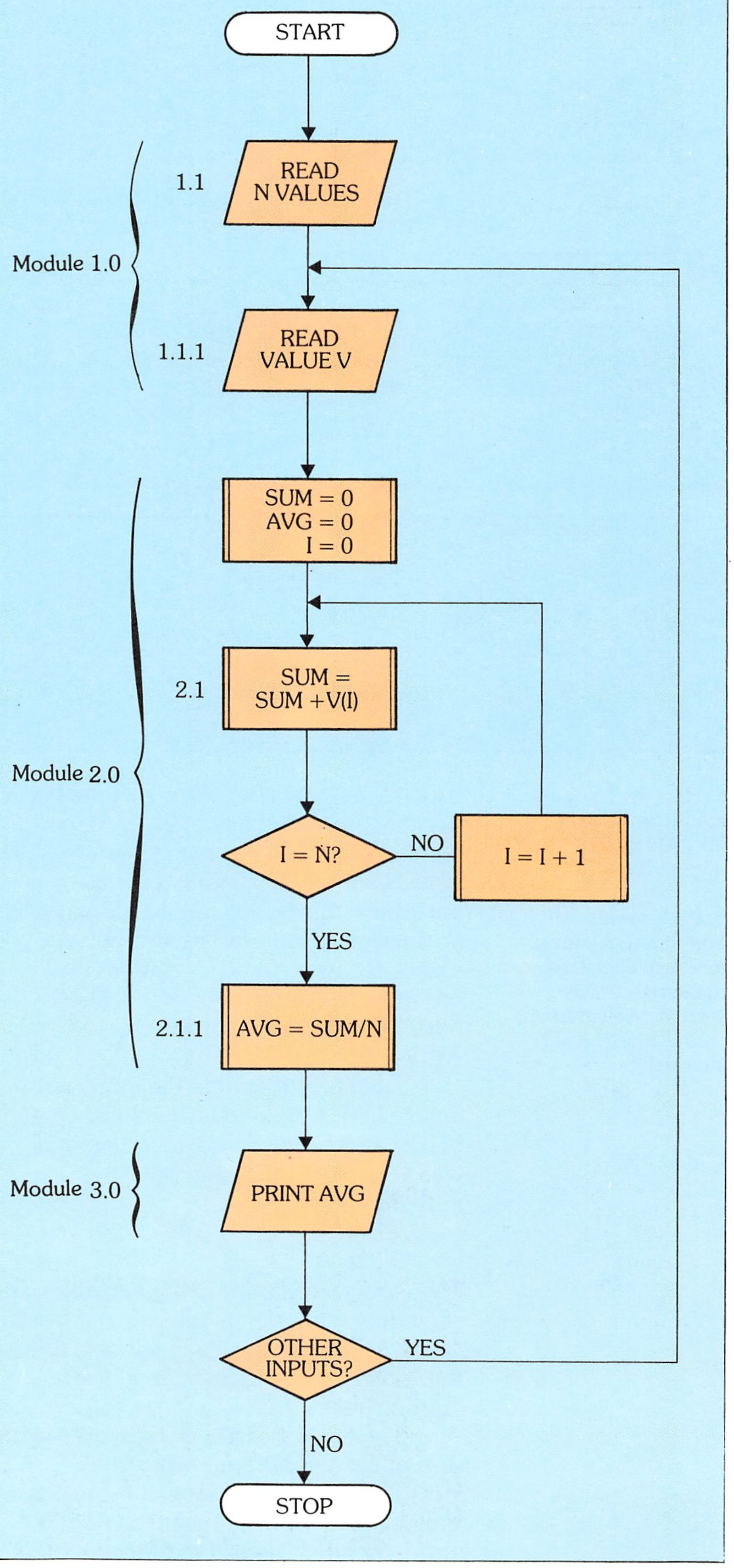
OTHER INPUTS?

If other students' data is input the answer is YES and the program enters a loop which sums and averages the grades until no further grades are input. At this point the decision answer is NO, and the loop is broken. In this way a simultaneous printed output can be made of all students' average grades.

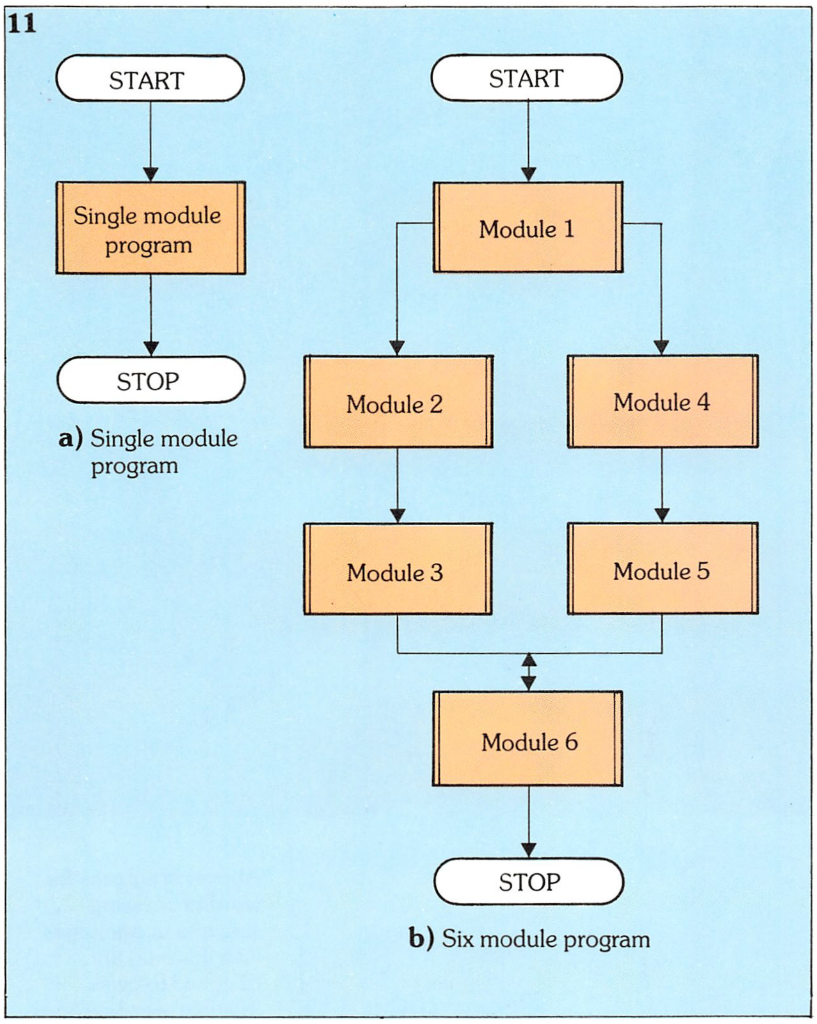
We can now accurately define the processes used in computer programming. A program is a number of computer instructions having a definite beginning and end. All the required instructions to get the computer through from beginning to end are contained within it.

A module is a part of a program which carries out a well defined function, but which is designed to operate with other interconnected program parts to form the complete program. *Figure 11a* shows an example of a single module program and *figure 11b* shows a possible six-module program. There is nothing which prevents the use of a module in more than one position in a program: modules 1, 3 and 5 of the program in *figure 11b* for example,

10



11



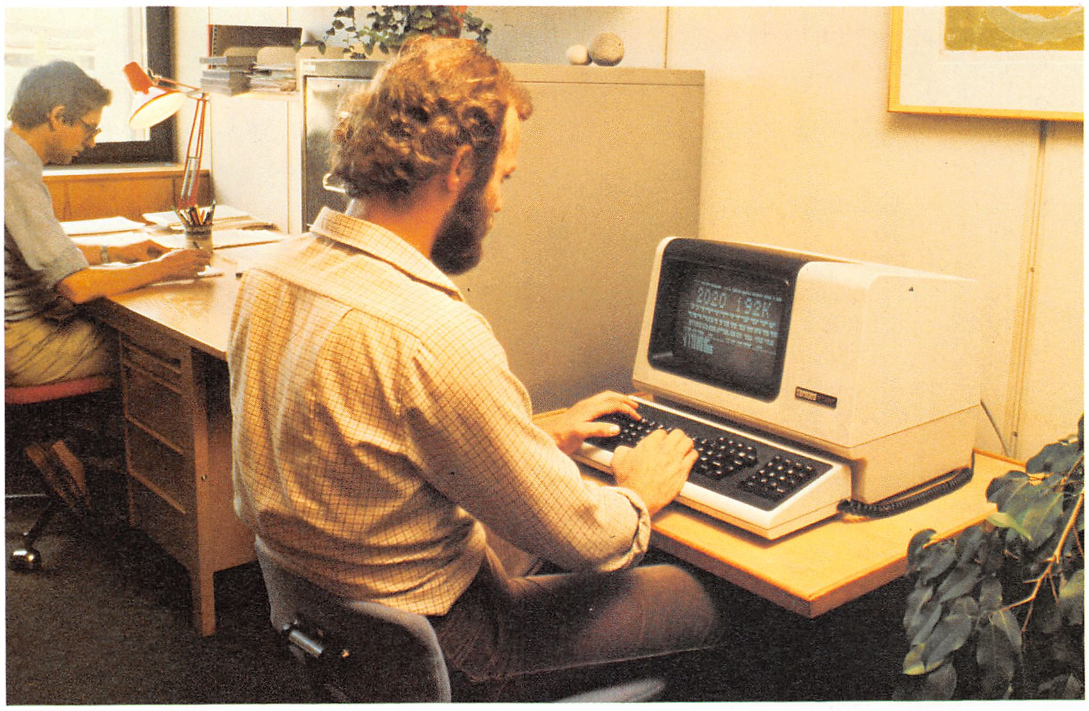
may be identical. Since different modules can work on the same variables, the output from one module can form the input to the next.

A **sub-routine**, like a sub-program, is part of a program that carries out a specific function when requested by the program. However, it differs from a sub-program in that it is not specific to the application and is usually called from a **library** of such sub-routines. For example, a multiplication sub-routine could be useful in a number of different programs for different applications. It will never work alone and is only called into operation when needed. This can be seen in *figure 12*, where a program makes use of two sub-routines. Which sub-routine is used, and when it is used, depends on what happens in the modules of the main block of the program (i.e. modules 1, 2 and 3). When a sub-routine process has been carried out, the instructions of the main block are then, generally, returned to.

These definitions are, in fact, a direct consequence of the use of top-down program design. Looking at the whole problem first (at the top-level) means that we can divide it up into smaller and smaller modules or sub-routines which can be duplicated if required and built into a complete **structured program**.

10. Flowchart for program determining the average grades for each of the six students.

11. (a) A single module program; (b) a six module program.



Right: it is good programming practice to type in your programme only after all design, writing and desk-checking is complete.

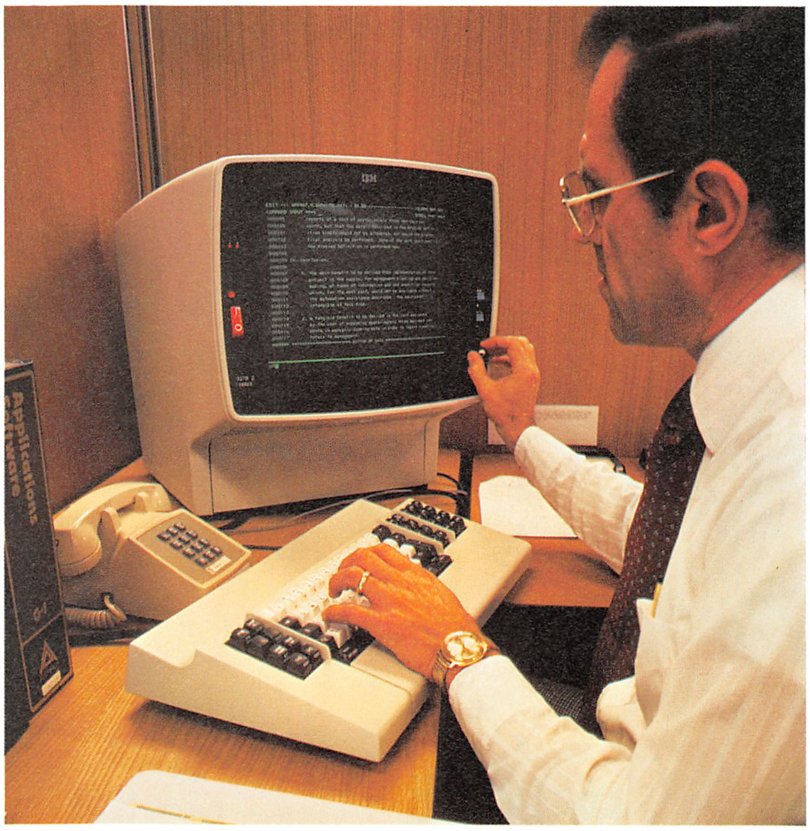
Tony Stone Associates

Translating the program

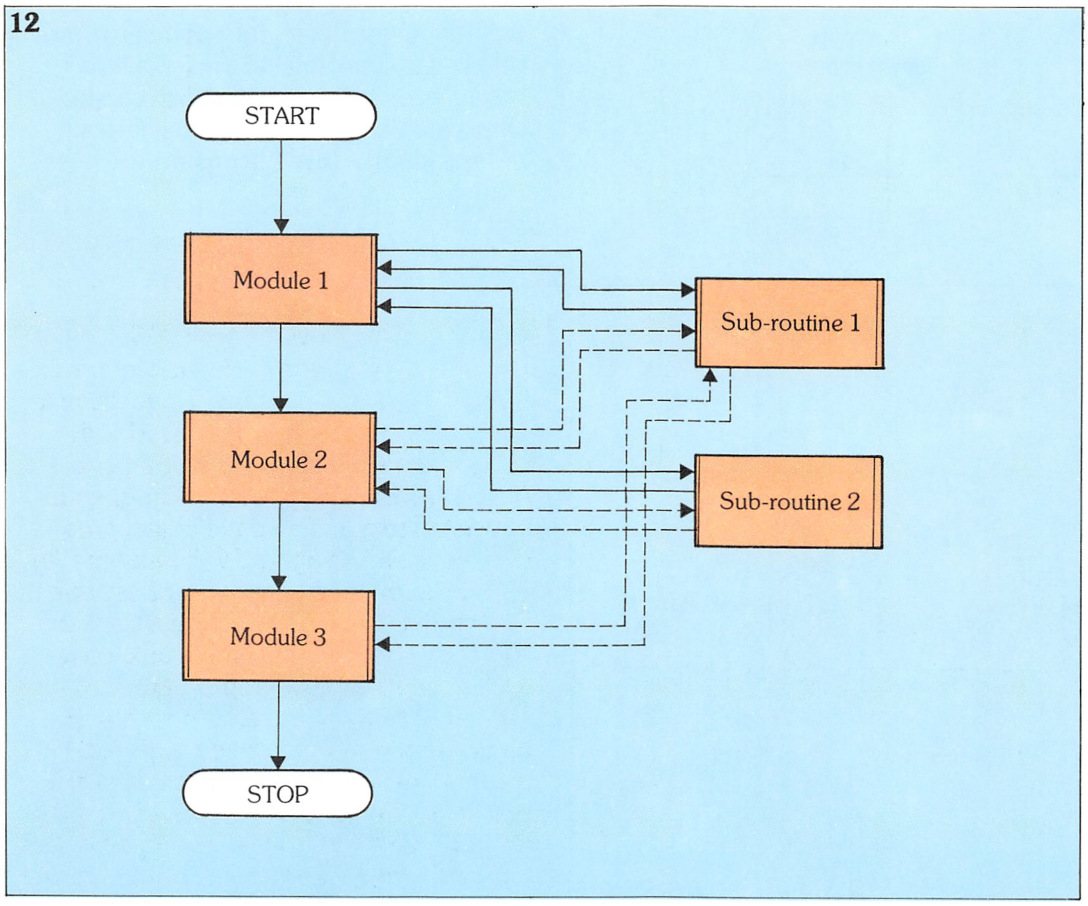
In order to be understood by the computer, the instructions of the programming language need to be translated into the computer's machine code. (If the program itself has been written in machine code then this stage is not required.)

The process of translation is undertaken by a **translator program**. The code to be translated by the translator program is termed the **source program**, and is written in a **symbolic language**, i.e. a language at a higher level than machine code. The result of the translation of the source program is the **object program** in machine code.

Translator programs are classified according to the nature of the source programming language to be translated. The level higher than machine code is assembly code which uses alphanumeric instruction statements, often abbreviated, which are known as **mnemonics**. A source program written in assembly language is translated into machine code by an

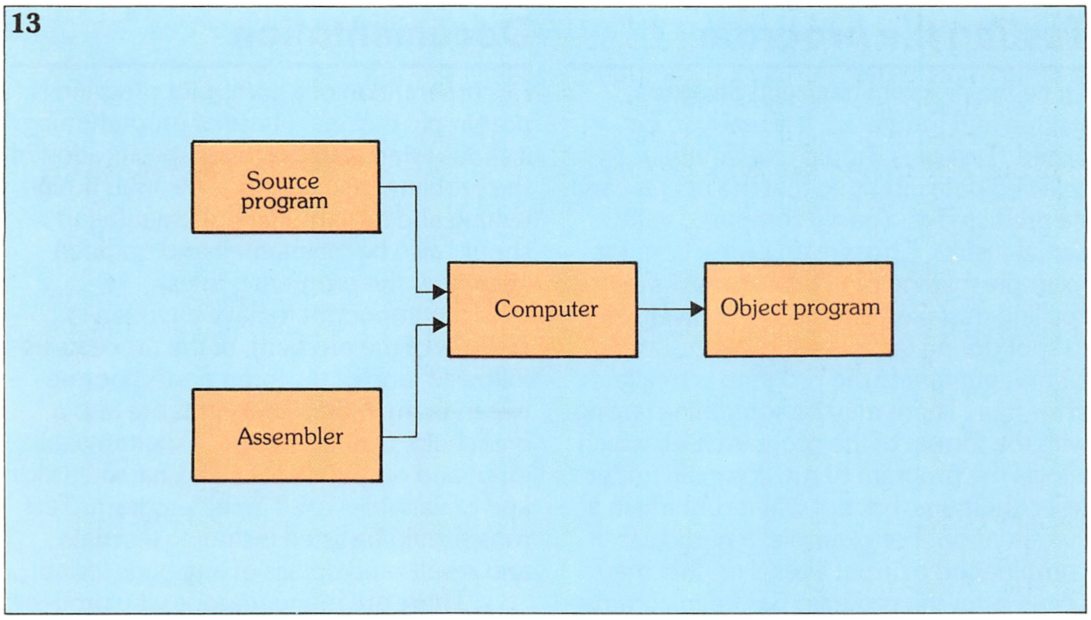

Tony Stone Associates

Above: programs for word processing systems or packages (like this one) are designed to be as 'friendly' as possible so that non-computer personnel can use them without difficulty.

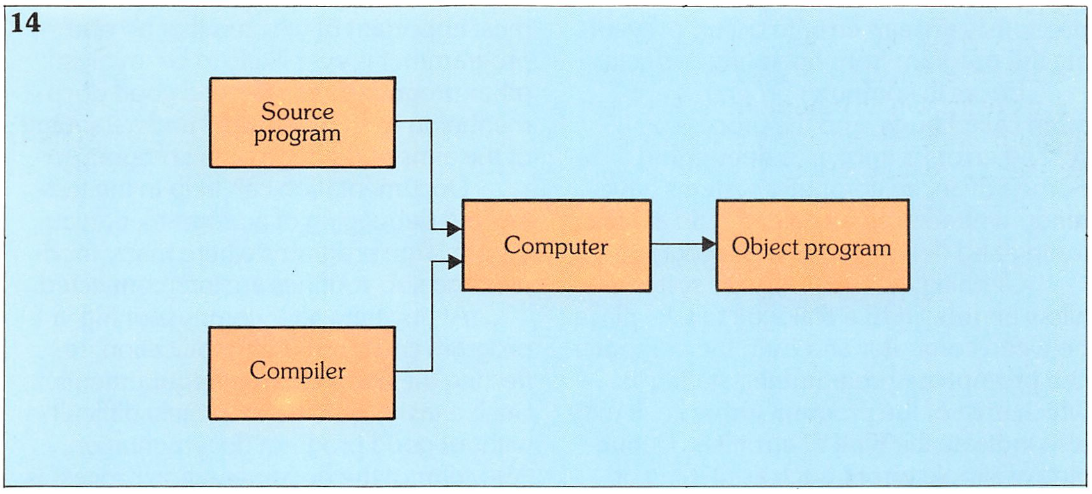


12. Sub-routines are called from a library to perform specific tasks – control then returns to the main program.

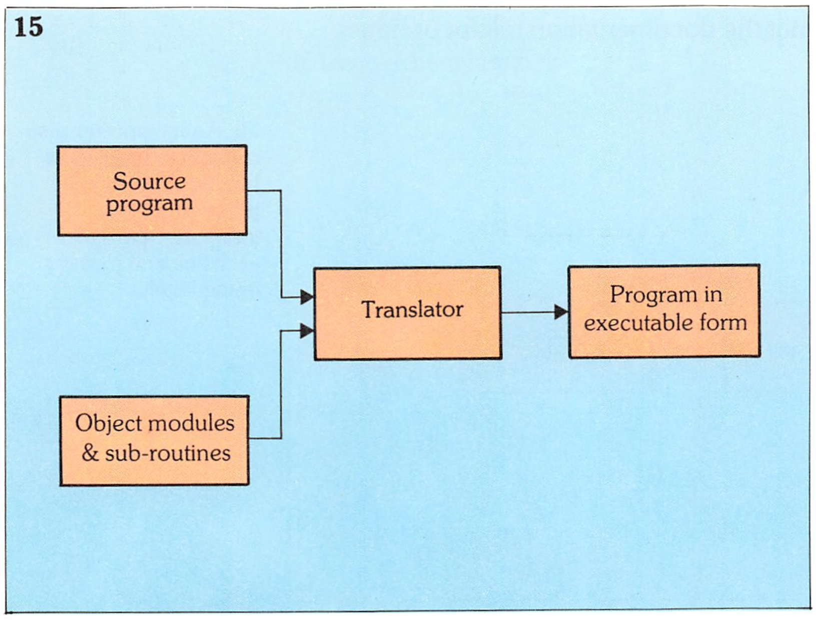
13. A source program written in assembly language is translated into machine code by an assembler.



14. A high-level language source program is translated into machine code by a compiler.



15. Assemblers and compilers translate the source program into a usable object program.



assembler, as shown in *figure 13*.

These mnemonics, however, are quite difficult to use, so even higher-level programming languages are used with instructions similar to everyday language. A high-level language source program is translated to machine code object program, as shown in *figure 14*, by a **compiler**.

Both of the examples of translation in *figures 13* and *14* show how assemblers and compilers translate the source program into a usable object program, as illustrated in *figure 15*. Another type of translator for high-level languages, called the **interpreter**, translates each line and runs it immediately (see *figure 16*). Here, the computer 'talks back' (prompts) where required; this is often the case with personal computers.

Testing the program

Once the program has been designed, written and translated, it is ready to be tested. Test runs should use carefully selected data inputs so that the results can be predicted and easily compared with actual results. Errors in an instruction, for example missing punctuation marks or spelling mistakes, should be relatively easy to spot during test runs. However, testing cannot *guarantee* the program is totally error free. There may be something wrong with the format of the program itself which allows the program to run correctly under test conditions, but not when running in a live situation. For example, a particular combination of input events or data may occur, after the program has been commissioned (i.e. put into use), which causes a previously unseen error to occur, preventing the program from operating correctly.

Errors in computer programs are often called **bugs**, and the process of solving errors is known as **debugging**. Some advanced computer systems have ready-written programs which can aid the testing and debugging of new programs.

Certain types of computer systems allow an **interactive** dialogue to take place between computer and user: the computer can prompt the programmer, stating in which area of the program errors lie. It may also indicate the type of error it is. Debugging of simple errors, such as missing or incorrect punctuation marks, can be made much simpler if interactive computer systems are used.

Documentation

Documentation of a computer program is not simply the final phase of programming: it should start with the initial specification of the problem, and continue through design, writing and testing. The documentation should also be maintained and updated even after the program is in use.

It is important to have a written analysis of the problem, of the procedures followed, and of the interconnections between them. Good flowcharts are also a great help. Details should be written of all input and output data, of file characteristics and of variables used in the program. Test runs should be listed including test data and results and details of any bugs found.

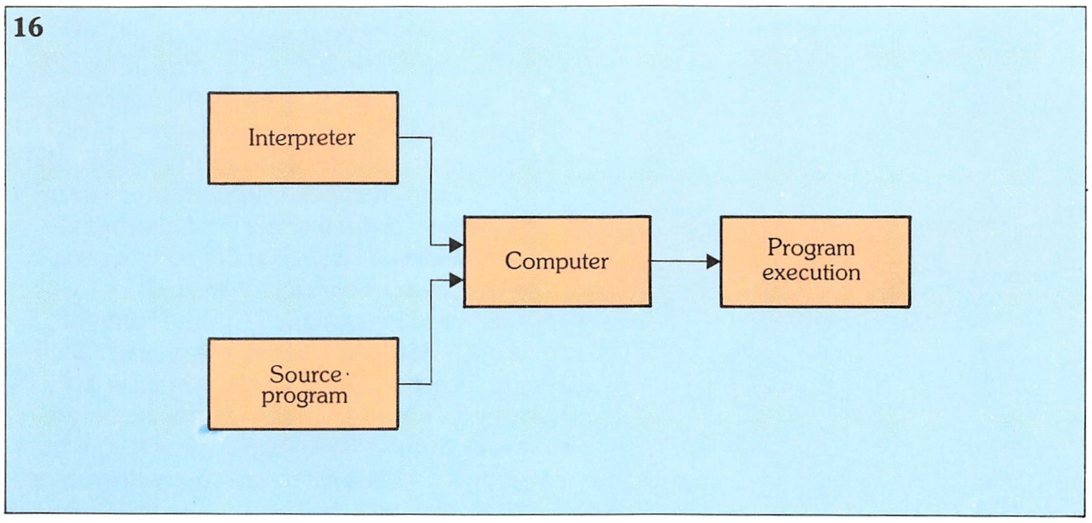
There are many reasons why programs should be properly documented. The most important of which is that no one programmer is ever likely to be involved; other programmers will need good documentation to aid their rapid understanding of the aims and design of the program.

Documentation can help in the testing and debugging of a program, particularly in large programs where many modules and sub-routines are interconnected.

At any time after commissioning, a program could require modification, reflecting the users different requirements. Such a task would be extremely difficult without good program documentation. Also, part of the documentation process is to include details of any modification so that the documentation is kept updated.

(continued in part 15)

16



16. An interpreter also translates high-level language source programs – the interpreter translates each line and runs it immediately.